



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TIIA LEUHU
SENTIMENT ANALYSIS USING MACHINE LEARNING

Master's thesis

Examiner: University lecturer Heikki
Huttunen Examiner and thesis ap-
proved by the Academic Board on
21 April 2014

ABSTRACT

TIIA LEUHU: Sentiment analysis using machine learning
Tampere University of Technology
Master of Science Thesis, 54 pages, 0 Appendix pages
May 2015
Master's Degree Programme in Signal Processing
Major: Multimedia
Examiner: university lecturer Heikki Huttunen

Keywords: sentiment analysis, supervised learning, pattern recognition, data mining, naive bayes, random forest, k-nearest neighbor, twitter

In recent years, social media and TV-production has formed a strong link between each other. The most popular social media platform in TV-industry is Twitter, where over a million tweets are shared in one day. Tweet content is feedback straight from the viewers, and might include more valuable information than individual surveys. Going through millions of tweets is hard or impossible manually. This thesis studies, how to teach a machine by supervised manner to analyze tweets. Machine analyzes sentiments based on the features that tweets include.

The main goal of this thesis is to clarify how the content can be received, prepared, extracted and classified. The study indicates that sentiments can be caught from Twitter data using mathematical patterns.

The thesis is divided into 5 chapters. Chapter 1 is the introduction for the sentiment analyzing with machine learning capabilities. Chapter 2 is the literature study part, where elements and techniques are explored. Chapter 3 is the implementation part, where selected classification methods and techniques for text data are specified. Chapter 4 covers results and chapter 5 finishes the work with conclusions.

TIIVISTELMÄ

TIIA LEUHU: Tunneanalyysi koneoppimisen avulla

Tampereen teknillinen yliopisto

Diplomityö, 54 sivua, 0 liitesivua

Maaliskuu 2015

Signaalinkäsittelyn diplomi-insinöörin tutkinto-ohjelma

Pääaine: Multimedia

Tarkastaja: yliopiston lehtori Heikki Huttunen

Avainsanat: tunneanalyysi, ohjattu oppiminen, hahmontunnistus, tiedon louhinta, naive bayes, random forest, k-nearest neighbor, twitter

Viime vuosina sosiaalinen media ja TV-tuotanto ovat muodostaneet vahvan linkin toisensa välille. Suosituin media-alusta keskustelulle TV-tuotannossa on Twitter, missä vaihdetaan päivittäin yli miljoona twiittiä. Twiittien sisältö on suoraa palautetta katsojilta, joiden sisältö voi olla arvokkaampaa kuin yksittäiset mielipidekyselyt. Miljoonien twiittien manuaalinen läpikäyminen on vaikeaa tai lähes mahdotonta. Tämä diplomityö tutkii, kuinka opettaa kone analysoimaan valvotusti twiittejä. Kone analysoi twiittien sisältämien piirteiden avulla, millaisia tunnetiloja twiitit sisältävät.

Diplomityön tavoite on selvittää kuinka saadaan hankittua datasisältöä, kuinka se esikäsittellään, irrotetaan ja luokitetaan. Tutkimus osoittaa, että tunnetila voidaan irrottaa Twitter datasta käyttämällä matemaattisia kaavoja.

Diplomityö on jaettu 5 osaan. Kappale 1 sisältää johdantoa koneoppimisen mahdollistamaan tunneanalyysiin. Kappale 2 on kirjallisuusosio, jossa käydään läpi elementit ja tekniikat. Kappale 3 on toteutuskappale, jossa avataan valitut tekniikat ja toimintamalli toimivan analysaattorin rakentamiseen. Kappale 4 pitää sisällään työn tulokset sekä analysoinnin ja kappale 5 päättää työn tiivistelmään.

PREFACE

Studying at Technical University of Tampere has offered a lot. Multimedia as a major was the right choice and even though all the hard work, I have enjoyed every bit of it. This thesis is written towards the interest of pattern recognition methods, machine learning and improving interactivity between TV-content and viewers.

I would like to say thank you for all supportive people around me during these four years. Thanks go also for Demola, Eyeworks, and Heikki Huttunen.

Now it is time for a self-five.

Tampere, 20.05.2015

Tiia Leuhu

CONTENTS

1. INTRODUCTION	1
2. THEORY	5
2.1 Feature extraction from text	8
2.1.1 Word count features	8
2.1.2 TF-IDF features	10
2.2 Feature selection	12
2.2.1 Select k-Best	13
2.2.2 Forward Selection	14
2.2.3 Recursive feature elimination	15
2.2.4 LASSO	16
2.3 Classification	17
2.3.1 Random forest	20
2.3.2 K-Nearest Neighbor	22
2.3.3 Naïve Bayes	23
2.3.4 Comparison between above algorithms	25
2.4 Validation / Evaluation	26
3. IMPLEMENTATIONS	28
3.1 Data retrieval	29
3.2 Lexicon based sentiment classification	32
3.3 Learning based sentiment classification	33
3.4 Feature extraction	34
3.5 Feature selection	35
3.6 Classification	38
3.6.1 Random forest	41
3.6.2 K-Nearest Neighbor	42
3.6.3 Naive Bayes	43
4. RESULTS	44
5. CONCLUSIONS	51
REFERENCES	53

LIST OF SYMBOLS AND ABBREVIATIONS

Accuracy	the proportion of true results (both true positives and true negatives) among the total number of cases examined
API	application programming interface
Bagging	the essential idea in bagging is to average noisy but approximately unbiased models, and hence reduces the variance, see bootstrap aggregating
Bias	error from erroneous assumptions in the learning algorithm
Bootstrap aggregating	also called bagging, a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression
Confusion matrix	a specific table layout that allows visualization of the performance of an algorithm, used typically in supervised learning
Corpus	text corpus (in linguistics), a large and structured set of texts
Eyeworks	a production company behind SuomiLOVE
Feature vector	an n-dimensional vector of numerical features that represent some object
Finnish	a part of the Finno-Ugric branch of the Uralic language family
Fisher's criterion	a classification method that projects high-dimensional data onto a line and performs classification in this one-dimensional space where the projection maximizes the distance between the means of the two classes while minimizing the variance within each class
Hyper surface i.e.	a generalization of the concept of hyper plane identically distributed
IDE	an integrated development environment
Logistic regression	type of probabilistic statistical classification model
OAuth	used to connect users to Twitter and sending secure, authorized requests to the Twitter API
Over fitting	happens when a machine learning algorithm captures the noise of the data
Precision	the proportion of the true positives against the entire positive results (both true positives and false positives)
Sparse matrices	used in arithmetic operations: they support addition, subtraction, multiplication, division, and matrix power
SuomiLOVE	TV-format where 100 love stories are told by a 100 love songs
Twitter	micro blogging service that allows people to communicate with short 140-character messages that roughly correspond to thoughts or ideas
Under fitting	happens when high bias causes an algorithm to miss the relevant relations between features and target outputs
Valid	a measurement system is valid if it is both accurate and precise
Variance	error from sensitivity to small fluctuations in the training set. High variance can cause over fitting: modeling the random noise in the training data, rather than the intended outputs.
#	the # symbol, called a hashtag, is used to mark keywords or topics in a Tweet.

1. INTRODUCTION

TV culture has changed during last year's when social media has become an important part of it. Most of the discussion takes place on social networks and TV companies want to make use of it. When adding interactivity between TV and viewers, social media forms a dialogical connection. Getting live feedback from the audience is valuable and it should be exploited somehow. One alternative is to analyze sentiments from received social media texts. For example, by mining Twitter data, *tweets*, feelings and opinions about the live TV content can be extracted. Results can be then expressed in graphics and numbers in real time.

The reason for analyzing and discovering knowledge from social media texts lies behind the big data revolution. Data streams flow fast and people have the ability and tools to analyze the content. Discovering knowledge from the data is also called data mining. Goal of the data mining is to extract information from a data set using machine learning algorithms. Extracted information is then analyzed and transformed into valuable and understandable form. In today's business environment, data mining possess great importance.

People use social media for different reasons. Whether the reason is that they want to be heard or satisfy the curiosity, they are using it. In 2014 there were 64 000 registered Finnish users in Twitter and 40% of Finnish citizens use Facebook. People want to write and read updates in social media easily and at once. It is a way to connect and engage with other people, share ideas, observations and experiences, ask questions, to be heard, and feel importance. People are curious about the world and how to organize and manipulate it.

Because of the need and interest towards sentiment analysis, Demola Tampere started a project where the goal was to create an analysator for tweets. Project started by implementing a sentiment analysis with a lexical approach. The lexical project was implemented for YLE where a football match between Finland and Hungary was analyzed. Due to certain factors, the project continued later under different circumstances and approach for sentiment analysis changed to learning based method. Sentient analysator using machine learning algorithms was for a TV-show called SuomiLOVE.

SuomiLOVE was a new kind of TV format where 100 love stories are told by a 100 love songs. Love is not just about romantic, thus stories tell about friendship, family or being a fan. Stories include winning the fears, unbelievable luck, tough counting on life, deep grief and great joy. SuomiLOVE goes deep and opens tear channels, but also

brings happiness and laugh. Songs in stories varies from classics to newest hits. The mission was to analyze what kind of emotions people feel when watching the show and whom artist, story or song gets the most feelings and tweets.

As a social media platform Twitter allows you to keep up with the latest happening of any other user even when you don't know the other user even exist. This kind of media channel suits well for TV and for opinion mining. Some social media platforms like Facebook and LinkedIn require the mutual acceptance of a connection between users. Twitter gives boundless opportunities to satisfy human's curiosity. When analyzing, tweet must contain a hashtag or a word, which allows grouping and searching similar messages.

Hashtag became a style for Twitter posts during 2009-2010 and has been used in a mass broadcast media promoting, purchasing, event promotion, consumer complaints and sentiment analysis. Hashtags reveals the sentiment an author attaches to a statement. It can be the state of mind, statement to make a tweet more powerful or to make it sarcastic. Nowadays almost every popular social media platform uses hashtags in their service.

Sentiment analysis from tweets (text data) is a natural language processing (NLP) task that includes pointing out the writers feeling about products, services or specific topic. A sentiment analysis determines the response of a user or a group of users on a topic and categorizes opinions as positive, negative or neutral. Natural Language Processing means computer manipulation of natural language. NLP is also known by the name of Computational Linguistics. As simplest NLP can be, counting word frequencies to compare writing styles or in more complex way, understanding complete human utterances. NLP in sentiment analysis can be done for example using Natural language Toolkit (NLTK).

Analyzing sentiment from the tweets has been done before this work. Most of them are done in English and there was no Finnish sentiment analysis tool for this kind of purpose when the project started. Today, there are few Finnish companies offering sentiment analysis for Twitter data. Researches and documents about data mining, sentiment analysis and machine learning are available in great amount, and this thesis delves into decision to make analysis using scikit-learn and NLTK with Python. Different techniques and algorithms to make a sentiment analysis were tested during the project and the best working algorithms were chosen for the analysator.

Sentiment analysis can be done in three ways. One is a lexicon-based technique where a dictionary is used to perform entity-level sentiment analysis. This technique uses dictionaries of words annotated with their semantic orientation (polarity and strength) and calculates a score for the polarity of the document. Usually method gives high precision but low recall. The second way to do sentiment analysis is a learning based technique,

which requires creating a model by training the classifier with labeled examples. This means gathering a dataset with examples for each class, extract features/words from the examples and then train the algorithm based on examples. The third method for sentiment analysis is a linguistic analysis, which in contrast, exploits the grammatical structure of text to predict its polarity, often in conjunction with a lexicon. For instance, linguistic algorithms may attempt to identify context, negations, superlatives and idioms as part of the polarity prediction process [15].

Choosing the right method depends on the application, domain and language. Lexicon based techniques enables achieving good results when using large dictionaries. Learning based techniques deliver good results when obtaining data sets and training. Main difference selecting to use a statistical technique or a syntactic one is following: Syntactic technique uses rules of the language in order to detect the verbs, adjectives and nouns. Syntactic technique may achieve better accuracy, but is heavily depending on the language of the document and the classifiers can't be ported to other language. Statistical techniques use probabilistic background and focus on the relations between the words and categories. When comparing statistical and syntactic techniques, statistical method benefits over the syntactic ones, by making translation into another language easy. Using statistical technique in other language is possible with minor or with no adaptations. Quite good results can be achieved when using machine translation of the original data set [17].

When making analysator in Finnish, straight converting from English to Finnish is impossible. In Finnish language, there are 200 morphological derivations in linguistics and words can mean two or even more things. For example, word '*pöllö*' means an *owl* and *stupid*, word '*kuusi*' means number *six* and a *spruce*. The Finnish alphabet is based on the same Latin alphabet used in English, plus three vowels with diacritics which are placed after z, *å*, *ä*, *ö*. Grammar between English and Finnish has some same basics, for example Subject-Verb-Object word order, but in Finnish, it allows much more flexibility in the placement of elements in a sentence. Two further areas of difference result in negative transfer. In Finnish, there are no separate pronouns for *he* and *she*, and Finnish does not use the definite or indefinite article.

What comes to vocabulary, even though the languages have the same letters, there are no cognates since the languages are from distinct language families. Even words that are imported into Finnish are transcribed so that they lose their familiarity. For example, the English word *crazy* becomes '*kreisi*' in Finnish. Making vocabularies with different values is time spending even though vocabularies are available. This is one of the reasons that machine learning algorithms were used in the final version of the analysator. Either way, in the beginning, manual analyzing and labeling the tweets is mandatory.

One of the difficult things in any language is sarcasm. Even some humans can't understand it so how to teach a machine to be aware of sarcasm.

Automatic recognition, description, classification, and grouping of patterns are important problems in this work. A pattern has been defined [19] as “opposite of chaos; it is an entity, vaguely defined and that could be given a name”. For example, in this work a pattern is the way how a tweet is written and what kind of emotions it includes. Given a pattern, its recognition/classification may consist of supervised, unsupervised or semi-supervised classification. This study concentrates to supervised learning method and the best algorithms for text (tweets) analyzing in Finnish.

Thesis consists of 5 main chapters starting with an introduction. Section 2 covers the theory part of the work including feature extraction, feature selection and classification. Section 3 covers the implementation part, where the chosen methods are explained and demonstrated with a help of a block diagram. Section 4 covers the results and the final chapter number 5 covers discussion and conclusions.

2. THEORY

This section describes the theoretical principles and methodologies used in the implemented text classification. The objective of theory section is to designate the most relevant factors behind the machine learning when making sentiment analysis from text data. Machine learning in general investigates how computers can learn based on data and this is done so that people could more efficiently use available data. The machine needs to learn how to automatically recognize even complex patterns and make intelligent decisions based on data. In text classification task, this involves feature extraction, feature selection and classification.

Before going into text classification process, we look into knowledge discovery process, demonstrated in Figure 1 [4] as an iterative sequence. First process in knowledge discovery is data cleaning, where the noisy and inconsistent data is removed. After data cleaning, only usable data material should be left. Data integrations can be done by combining multiple data sources. When having a combined data mass, data selection can be performed, where the relevant data for the analysis is retrieved from the database. For the selected data, a data transformation is performed, where the data will be transformed and consolidated into appropriate forms for mining.

The essential part of the process follows data selection. This is called data mining, where intelligent methods are applied to extract data patterns. After discovering patterns, a pattern needs evaluation. The purpose is to identify the truly interesting patterns representing knowledge based on interestingness measures. At the end of the process, a knowledge presentation can be visualized and use knowledge representation techniques to represent mined knowledge for users.

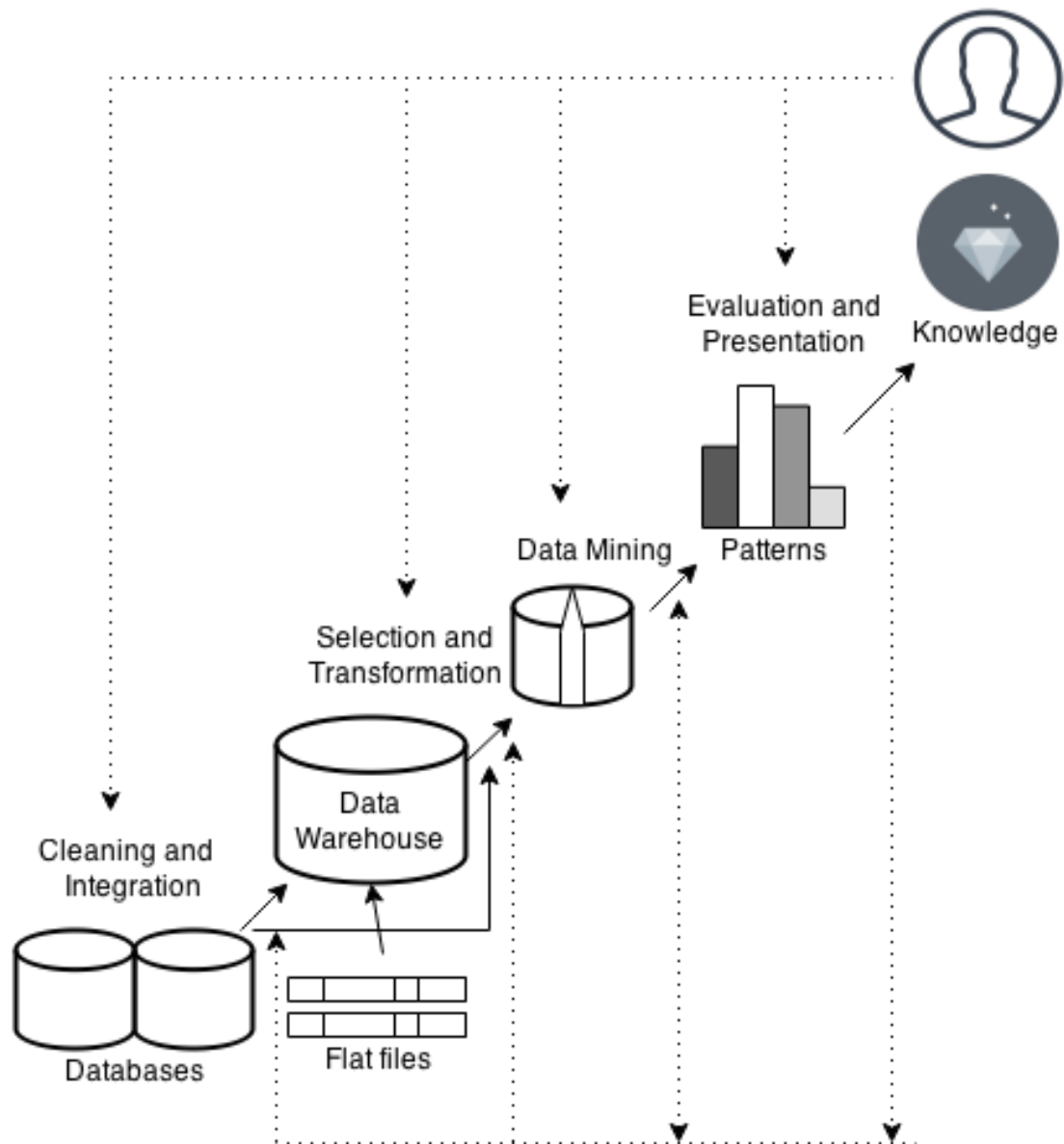


Figure 1. Steps to mine knowledge from the data. [4]

The scope of this thesis concentrates on the data mining step, where studying the text classification process: how to divide text segments to categories automatically. The text classification pipeline is illustrated in Figure 2, where the recognition system for text and images is split into two modes: training (learning) and classification (testing). In the training mode, the feature extraction and selection modules find the appropriate features for representing the input patterns and the classifier is trained to partition the feature space. The feedback path allows a designer to optimize the preprocessing and feature extraction and selection strategies. In the classification mode, the trained classifier assigns the input pattern to one of the pattern classes under consideration based on the measured features.

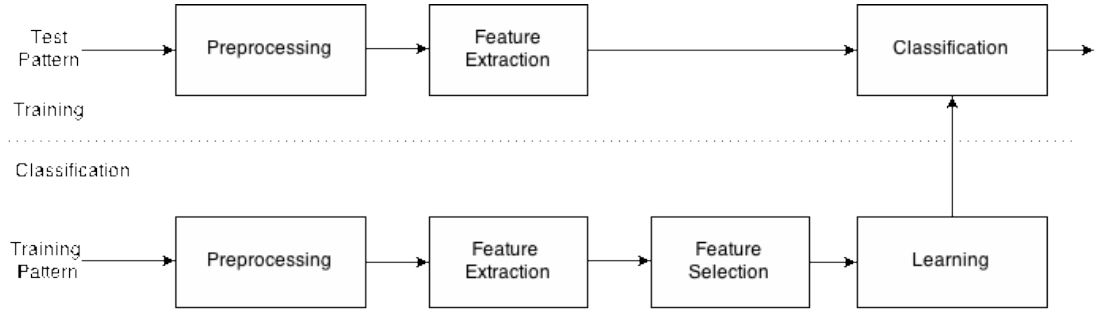


Figure 2. *Model for statistical pattern recognition.*

Distinction between feature selection and extraction is important. Feature extraction is related to dimension reduction, where the large input data to an algorithm is transformed into a reduced set of features. The term feature selection refers to algorithms that select the best subset of the input feature set. In the literature, these two terms are used interchangeably, but means different stages of the classification process. Feature extraction normally precedes feature selection, because first, features are extracted from the data and then some of the extracted features with low discrimination ability are discarded.

A term statistical pattern recognition can be used to cover all stages from problem formulation and data collection through to classification, assessment of results and interpretation. The decision making process in statistical pattern recognition can be summarized as follows: A given pattern is to be assigned to one of c categories $1, 2, \dots, c$ based on a vector of d feature values $x = (x_1, x_2, \dots, x_d)$. The features are assumed to have a probability density or mass (depending on whether the features are continuous or discrete) function conditioned on the pattern class.

Each pattern is a point in a d -dimensional space. Goal is to choose features that allow pattern vectors belonging to different categories to occupy compact and disjoint regions in a d -dimensional feature space. The goal is to make the feature set effective, so that patterns from different classes can be separated. From each class a set of training patterns is needed, because the objective is to establish decision boundaries in the feature space with separate patterns belonging to different classes.

The decision boundaries are usually determined by the probability distributions of the patterns belonging to each class, which must either be specified or learned. In statistical classification decision boundary is a hyper surface that partitions the vector space into number of sets (number of classes), one for each class. Decision boundaries for classification can be outlined also using a discriminant analysis-based approach. In this approach, a parametric form of the decision boundary is specified and then the best decision boundary of the specified form is found based on the classification of training patterns. This kind of boundaries can be constructed using for example some loss function to be minimized.

Assumptions about the nature of the training data can be very general or weak by most machine learning algorithms. It is typical that they require large amounts of training data to learn accurate classifiers. This problem can be solved by exploiting prior knowledge to eliminate from consideration classifiers that are not consistent with the prior knowledge [2]. This leads to learning algorithms that may be able to learn from very few training examples. It should be remembered that introducing prior knowledge involves a risk. If using incorrect knowledge, all accurate classifiers will be eliminated from consideration by the learning algorithm. Prior knowledge introduces bias into the learning process, and it is important that this bias is correct.

2.1 Feature extraction from text

Feature extraction consists of transforming arbitrary data, such as text or images, into numerical features usable for machine learning. Feature extraction starts with a dimensionality reduction to an initial set of measured data by building derived values about features that are informative and non-redundant. When the algorithm receives too large input data, that is suspected to be redundant, it can be transformed into a reduced set of features, so called feature vectors. This is the process of feature extraction and can be called vectorization. Extracted features contain the relevant information from the input data and will be used for further means instead of the complete initial data.

Feature extraction can be done for text and image data sets, in a format supported by machine learning algorithms. Because feature extraction reduces the amount of resources required to describe a large data set, performing analysis of complex data contains some problems. For example, when having a large number of variables, algorithm requires a large amount of memory and computation power.

2.1.1 Word count features

Typically, machine-learning algorithms are defined in terms of numerical vectors. This is because the raw data, a sequence of symbols cannot be fed directly to the algorithms. There are different ways to extract numerical features from text content. One choice is to use Bag of Words representation, which is a specific strategy using tokenization, counting and normalization. In this representation, documents are described by word occurrences while ignoring completely the relative position information of the word in the document.

Bag of Words first tokenize the strings and gives an integer identification for each possible token, for example using white spaces as token separator. Then it counts the occurrences of tokens in each document and finally normalizes and weights importance of tokens that occur in the majority of samples or documents. In this scheme, each individual token occurrence frequency is treated as a feature. The vector including all of the token frequencies for a given document is considered a multivariate sample. Data for

training can be represented by a matrix with one row per document and one column per word occurring in the corpus.

For example, if we tokenize and count word occurrences of a text documents: ['this is my first tweet.' 'This is my second tweet.' 'And the third one.' 'Is this my first tweet?']. Each term found by the analyzer during the fit is assigned a unique integer index corresponding to a column in the resulting matrix. This gives us now ['and', 'first', 'is', 'my', 'one', 'second', 'the', 'third', 'this', 'tweet'].

Tokenization algorithm mentioned above needs several preliminary tests to find the best algorithmic configuration. When tokenizing text, one choice is to use n-grams framework. N-gram, (also called shingles) is a contiguous sequence of n items from a given sequence of text or speech. Items for n-gram can be in case of text, letters, words or base pairs. The number of n should not be too big. In sentiment analysis, using 2-grams or 3-grams increases the number of keyword combinations and can hurt the results. Decision to take multiple occurrences of the words into account, it should be remembered that the number of occurrences of the word in the text does not make much of a difference. Binarized versions of the algorithms perform better than the ones that use multiple occurrences [17].

If you want to form 2-grams, in other words, bigrams, you define n-gram range for the vectorizer. For example, unigram vectorizer would analyze sentence *Sentiment analysis is fun!* as: ['sentiment', 'analysis', 'is', 'fun'] but using bigram, the vectorizer gives you ['sentiment', 'analysis', 'is', 'fun', 'sentiment analysis', 'analysis is', 'is fun']. This can make a huge difference in some languages, like in Finnish. For example, when trying to solve, if a text includes an opinion from two possible choices the word just in front of another word can make a huge difference. In Finnish *'pärjää'* (cope) and *'kyllä pärjää'* means positive and *'ei pärjää'* negative opinion. For this reason we need to use bi-grams or tri-grams for analyzing and consider carefully which words to include to the stop words list.

When extracting features from text, there usually are also not relevant words included in the data set. These irrelevant words come from the tokenization algorithm, and should be erased from the models feature list, to make a model more efficient. This is done by using a string or list of stop words in feature extractor. There are ready-made lists available for some languages but not for all. Usually list includes words that are not relevant to classify. Stop-words can make a big difference in classification and should be think through carefully.

After the input documents are indexed and the initial word frequencies computed, transformations can be performed to summarize and aggregate the information that was extracted. Generally, the frequencies of a word or term reflect on how important or salient a word in each document is. Words that occur with great frequency are usually better

descriptors of the contents of specific document. However, the word counts themselves should not be assumed proportional to importance as descriptors of the documents. For example, if a word '*työkalu*' (tool) occurs once in a document *A*, and four times in a document *B*, it is not necessarily reasonable to conclude that this word is four times as important a descriptor of document *B* as compared to document *A*. A common transformation of the raw word frequency counts (*wf*) is to compute:

$$f(wf) = 1 + \log(wf), \text{ for } wf > 0.$$

This transformation will weaken the raw frequencies and their affect to the results of subsequent computations. A simpler transformation can be used to enumerate whether a term is used in a document. This simpler version is called binary frequency, defined by:

$$f(wf) = 1, \text{ for } wf > 0.$$

Here the resulting document-term matrix contains only *one's* and *zero's*, which indicate the presence or absence of the respective words. Document-term matrix describes the frequency of terms that occur in a collection of documents. However, like in previous transformation, this will also weaken the effect of the raw frequency counts on subsequent computations and analyses.

Another issue to consider more carefully and reflect in the indices used in further analyses is the relative document frequencies (*df*) of different words. For example, a word '*luulen*' (I think) may occur frequently in all documents, while another word such as '*rakastan*' (I love) may occur only in a few. A common and useful transformation that reflects both the specificity of words (document frequencies) and the overall frequencies of their occurrences (word frequencies) is inverse document frequency (for the *i*'th word and *j*'th document):

$$idf(i, j) = \begin{cases} 0 & \text{if } wf_{ij} = 0 \\ (1 + \log(wf_{ij})) \log \frac{N}{df_i} & \text{if } wf_{ij} \geq 1 \end{cases} \quad (1)$$

In above formula (1) *N* is the total number of documents; *df_i* is the document frequency for the *i*'th word.

2.1.2 TF-IDF features

In text classification, a text document may partially match many categories. Finding the best matching category for the text document can be done with the term frequency/inverse document frequency (TF-IDF) approach [13]. TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. TF-IDF has found good use in document classification and clustering as it originally is a term weighting scheme for information retrieval. In text classification, it weights

each word in the text document according to how unique it is. The TF-IDF value increases proportionally to the number of times a word appears in the document and captures the relevancy among words, text documents or categories.

Example of term frequency could be having a set of Finnish tweets from which wanting to determine, what tweet is most relevant to the query '*Aivan älyttömän hyvä*' (totally awesome). A simple approach is to first eliminate the tweets that do not contain all three words '*aivan*', '*älyttömän*', '*hyvä*', but still several tweets are left. To further distinguish them, we count the number of times each term occurs in each tweet and sum them all together. The number of times a term occurs in a document is called its term frequency. The term frequency $tf(t, d)$ is simple when choosing to use raw frequency of a term in a tweet. For example, the number of times that term t occurs in a tweet d . If we denote the raw frequency of t by $f(t, d)$, the simple tf scheme is $tf(t, d) = f(t, d)$.

If the tweet contains a common term for example '*aivan*' (totally), this will incorrectly emphasize tweets that happens to use word '*aivan*' more frequently, without giving enough weight to the more meaningful terms '*älyttömän*' (ridiculously) and '*hyvä*' (good). This is an example of inverse document frequency. The term '*aivan*' is not a good keyword to distinguish relevant and non-relevant tweets and terms, unlike the less common words. Hence, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and this increases the weight of terms that occur rarely.

The inverse document frequency is a measure of how much information the word provides. In other words, is the term rare or common in all tweets. This measure is logarithmically scaled fraction of the tweets that contain the word, and is obtained by dividing the total number of tweets by the number of tweets containing the term, and finally taking the logarithm of that:

$$idf(t, D) = \log \frac{N}{|\{d \in D: t \in d\}|}. \quad (2)$$

In above formula (2), N is the total number of documents in the corpus, denominator is the number of documents where the term t appears. If the term is not in the corpus, this will lead to a division-by-zero. For this reason, it is common to adjust the denominator to $1 + \text{denominator}$.

Using TF-IDF instead of raw frequencies of occurrence of a token in a document gives opportunity to scale down the impact of tokens that occur very frequently. Frequently occurring tokens are less informative than features occurring more rarely in the training corpus.

Hereby the TF-IDF is calculated as:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D). \quad (3)$$

The effect of this (3) is that terms with zero *idf*, i.e. which occur in all documents of a training set, will not be entirely ignored. High term frequency will give high weight in TF-IDF and low tweet frequency of the term in the whole collection of tweets. This means that weights tend to filter out common terms. Ratio inside the IDF's log function is always greater or equal to 1; the value of TF-IDF is greater or equal to 0. When a term appears in great amount of tweets, the ratio inside the logarithm approaches 1, and brings the IDF and TF-IDF closer to 0.

2.2 Feature selection

After feature extraction the next important step is feature selection, which is essential to successful data mining. It is one of the most important parts of the text data preprocessing and frequently most used techniques. Feature selection reduces the number of features and removes irrelevant, redundant, or noisy data. Selection process brings the immediate effects by improving the scalability, efficiency and accuracy of a text classifier. For example, feature selection is important for applications, which need speeding up a data mining algorithm, and improving mining performance such as predictive accuracy and result comprehensibility.

In text classification, the feature selection is the process of selecting a specific subset of the terms of the training set and using only them in the classification algorithm. For text classification the high dimensionality of the feature space is a major problem. For counting an optimal feature set, evaluation criterion can be used as a measure system. When the dimensionality of a domain expands, the number of features N increases. The main advantages for using feature selection algorithms are the facts that it reduces the dimension of the data, improves accuracy by removing noisy features and it makes the training faster.

A noisy feature increases the classification error on new data, when added to the document representation. For example, a rare term, which has no information about a class X , but all instances of this term happen to occur in class X documents in training set. This causes that the learning method might produce a classifier that misassigns test documents containing this word to class X . This kind of incorrect generalization from accidental property of the training set is called over fitting. One way to avoid over fitting is to use consequence feature selection.

When designing the classifier, if the used training sample count is small relative to the number of features, the performance of a classifier can degrade if adding more features. This is referred as the peaking phenomenon. A reduction in the number of features may lead to a loss in the discrimination power and lower the accuracy of the resulting recognition system. Without bias, classification is impossible. When choosing the features, remember to make choices carefully, since it is possible to make two patterns too similar by encoding them with a sufficiently large number of redundant features.

The driving force of the training procedure is however, the minimization of a criterion such as the apparent classification error. Selection methods can be compared using for example cross-validation. Cross-validation is a model validation technique that estimates how accurately a predictive model will perform in practice. The best algorithm is usually found by trial and error. Data mining applications involve usually thousands of features, so the computational requirement of a feature selection algorithm is important.

All feature selection algorithms can be represented in a space of characteristics according to the criteria of search organization (Org), generation of successor states (GS) and evaluation measures (J). This space encompasses the whole spectrum of possibilities for a feature selection algorithm. Algorithm 1 shows the general algorithm for feature selection.

```

Input:
2   S - data sample with features X, |X| = n
    J - evaluation measure to be maximized
4   GS - successor generation operator
Output:
6   L := Start_Point(X);
    Solution := {best of L according to J};
8   repeat
    L := Search_Strategy(L, GS(J), X);
10  X' := {best of L according to J};
    if J(X') ≥ J(Solution) or (J(X') = J(Solution)
12    and |X'| < |Solution|)
    then Solution := X';
14 until Stop(J, L)

```

Algorithm 1. *General Algorithm for feature selection.*

2.2.1 Select k-Best

K-best method is one of the simplest feature selection methods available. K-best method composes the best subset of k features of the k best features, which are considered one at a time. K-best method has its downside, because a set of the best individual k feature is not necessarily the best set of k features.

Univariate feature selection holds Select k-Best, and works by selecting the best features based on univariate statistical tests. This can be seen as a preprocessing step to an estimator. The basic feature selection algorithm for selecting the k best features is presented in algorithm 2 [11].

```

SELECTFEATURES(D,c,k)
1   V←D
2   L←[ ]
3   for each t ∈V
4   do A(t,c)←COMPUTEFEATUREUTILITY(D,t,c)
5   APPEND(L(A(t,c),t))
6   return FEATURESWITHLARGESTVALUES (L,k) )

```

Algorithm 2. *Feature selection algorithm for selecting the k best features.*

In algorithm 2 for a given class c , compute a utility measure $A(t, c)$ for each term of the vocabulary D and select the k terms that have the highest values of $A(t, c)$. All other terms are discarded and not used in classification. [11] There are three different utility measures, which works for text data. These utility measures are Mutual information, Chi square test, and frequency. Each feature selection algorithm evaluates the keywords in a different way and thus leads to different selections. In addition, each algorithm requires different configuration such as the level of statistical significance, the number of selected features et cetera.

2.2.2 Forward Selection

In Forward Selection, the best single feature is selected and then one feature at a time is added, which in combination with the selected features maximizes the criterion function. Once a feature is retained, it cannot be discarded. Forward selection is computationally attractive because to select a subset of size 2, it examines only $(d - 1)$ possible subsets.

Starting with $X' = \emptyset$

Adds features to the current solution X' , among those that have not been selected yet

In each step, the feature that make J be greater is added to the solutions

The cost of operator is $O(n)$.

Algorithm 3. *Forward Selection algorithm.*

In forward feature selection, all feature subsets, which consist of only one input attribute, are evaluated at the beginning. For example, in case of one-component subsets $\{X_1\}, \{X_2\}, \dots, \{X_M\}$, where M is the input dimensionality, measuring starts with the Leave-One-Out Cross Validation (LOOCV) error. Purpose of this measurement is to find the best individual feature $X(1)$. After LOOCV, forward selection finds the best subset consisting of two components $X(1)$ and feature from remaining $M - 1$ input attributes. It can assumed, that $X(2)$ is the next best pair. This continues evaluating the third, fourth and more features for the input subset. The best feature subset in forward selection is the one with m -tuple consisting of $X(1), X(2), \dots, X(m)$.

The best feature set is the winner out of all the M steps. If the cost of a LOOCV evaluation with I features is $C(i)$, the computational cost of forward selection searching for a feature subset of size m out of M total input attributes will be $MC(1) + (M - 1)C(2) + \dots + (M - m + 1)C(m)$. The overall best input feature set can be found also employing exhaustive search. Method begins with searching the best one component subset of the input features, which is the same in forward selection algorithm. Next, it goes to find the best two-component feature subset, which may consist of any pairs of the input features. Then it moves to best triple and so on. From the cost of exhaustive search it is clear that it is $MC(1) + (M/2) * (C(2) + \dots + (M/m) * C(m))$.

When comparing exhaustive search and forward selection, the latter is much cheaper. Forward selection can suffer from its greediness. For example, if $X(1)$ is the best individual feature, it does not guarantee that either $\{X(1), X(2)\}$ or $\{X(1), X(3)\}$ must be better than $\{X(2), X(3)\}$. For this reason, a forward selection algorithm may select a feature set different from that selected by exhaustive searching. If making a bad selection of the input features, the prediction \hat{Y}_q of a query $X_q = \{x_1, x_2, \dots, x_M\}$ is significantly different from the true Y_q .

2.2.3 Recursive feature elimination

Recursive feature elimination (RFE) is a robust and “brute-force” method where the impacts of combined features are evaluated together. RFE is done in backward stepwise manner, starting with the smallest weights and moving on to larger weights. In this method, a model is first trained with all the features and evaluated the performance on held out data. Then the weakest features are chosen and retrained on the remaining features. Iterating continues until a sharp drop in the predictive accuracy of the model can be seen.

Starting with $X' = X$

Removes features from the current solution X' , among those that have not been removed yet.

In each step, the feature that makes J be greater is removed from the solution.

The cost of operator is $O(n)$

$X' = X - \{x_i \in X' | J(X' - \{x_j\}) \text{ is bigger} \}$

Algorithm 4. *RFE algorithm.*

The goal of recursive feature elimination (Algorithm 4) is to select features recursively considering smaller and smaller sets of features. The initial set of features is trained and then weights are assigned to each one of them (or the coefficients of a linear model). After training, the features whose absolute weights are the smallest are pruned from the current set features. This procedure will be repeated recursively on the pruned set until the desired number of features to select is eventually reached. In addition to RFE, rank-

ing features of the best number of features can be done using RFECV (Recursive Feature Elimination and Cross-Validated).

2.2.4 LASSO

The LASSO (Least Absolute Shrinkage and Selection Operator) is also called L1 penalized linear method that estimates sparse coefficients. L1 regularizer promotes feature selection while learning, and is a considerable choice when training a generalized model for classification or regression. In this algorithm the coefficient for the weakest features are set to zero by the learning algorithm itself.

Despite that L1 is called as logistic regression; it is a liner model for classification rather than regression. Lasso is useful in some contexts due to its tendency to prefer solutions with fewer parameter values and effectively reducing the number of variables, which are needed for the solution. Lasso and its variants are fundamental to the field of compressed sensing and under certain conditions; it can recover the exact set of non-zero weights.

Mathematically the Lasso consists of a linear model, which is trained with l_1 prior as regularizer. L1 regularized regression solves the optimization problem following:

$$\min_{w,c} \|w\|_1 + C \sum_{i=1}^n \log \left(\exp \left(-y_i (X_i^T w + c) \right) + 1 \right).$$

The Lasso estimate solves the minimization of the least-squares penalty with $\alpha \|w\|_1$ added, where α is a constant and $\|w\|_1$ is the l_1 -norm of the parameter vector. The implementation in the class Lasso in scikit-learn uses coordinate descent as the algorithms to fit the coefficients.

For a good choice of α , the Lasso can recover fully the exact set of non-zero variables using only few observations. The number of samples should be large, or L1 model will perform at random and the definition of large depends on the number of non-zero coefficients, the logarithm of the number of features, the amount of noise, the smallest absolute value of non-zero coefficients, and the structure of the design matrix X . This design matrix X cannot be correlated and display certain specific properties. There is no general rule for selecting an α parameter for recovery of non-zero coefficients. α can be set by cross-validation LassoCV or LassoLarsCV, but this can lead to under-penalized models. Under-penalized models refers that including a small number of non-relevant variables is not detrimental to prediction score.

2.3 Classification

Classification can be expressed as a categorization process where objects are recognized, differentiated and understood. A data object represents an entity, typically described by attributes. Data objects become data tuples when stored in a database, where rows correspond to the attributes. Classification process finds a model that describes (discrete, unordered) data class labels. This model is made of on the analysis of a set of training data. Purpose of this model is to predict the class label of objects for which the class label is unknown. Classification is usually referred in machine learning to supervised learning. There are also unsupervised, semi-supervised and active learning methods available to perform classification. Many classification methods have been proposed by researchers in machine learning, pattern recognition, and statics.

While performing Sentiment analysis for text, the class labels must be think through carefully. If using polarity classes' positive and negative, also a neutral class needs to be taken into consideration. Not every comment on a product or experience expresses purely positive or negative sentiment. In many cases, some comments include objective facts without expressing any sentiment, while others might express mixed or conflicting sentiment [9]. Training the classifier to detect only the two classes forces several neutral words to be classified either as positive or negative, which leads to over fitting. Learning from negative and positive examples alone will not permit accurate classification of neutral examples. Moreover, the use of neutral training examples in learning facilitates better distinction between positive and negative examples [9].

Classes can refer to different things and can be numeral or textual. For example, they can indicate whether a sentence involve a specific word or not. In this case, classes are 'yes' (1) and 'no' (0). Another example is to point out what color should one paint their house. If answers indicate red, green, blue, yellow, and brown, these are the classes to use when classifying the answers. In text data sentiment analysis there can be classes for happiness, sadness, sarcasm, sports, love, hate and so on. One object can have more than one class if wanted. Marking classes more than one can help in further development of the analysator, when having already labeled examples for different classes.

Classification problem can be choosing between answers 'yes' or 'no', 'happy' or 'sad' or 'party A', 'party B', 'party C'. These categories can be represented by discrete values, where the ordering among values has no meaning. Data classification is a two-step process where the first process is the learning (constructing the classification model) and the second one a classification (predicting class labels for given data by the model).

Supervised learning means, that the supervision in the learning comes from the labeled examples in the training data set. For example handwritten postal codes images and their corresponding machine-readable translations are used as the training examples, which supervise the learning of the classification model. For supervised classification

and regression, there are many different learning algorithms available. These algorithm types are grouped according to the formalism they employ for representing the learned classifier or predictor. Groups are separated to decision trees, decision rules, neural networks, linear discriminant functions, Bayesian networks, support vector machines, and nearest-neighbor methods.

When the values for the class properties in the training set are unknown, it is unsupervised learning which refers for clustering. In this method, the input examples are not class labeled. Clustering can be used to find classes within the data, where the algorithm clusters the data into different groups, like recognizing different types of headlines. Semi-supervised learning is a class of machine learning techniques that make use of both labeled and unlabeled examples when learning a model [4]. One way to classify with semi-supervised learning is to use labeled examples to learn class models and use unlabeled examples to refine the boundaries between classes. When the user play an active role in the learning process, it is active learning. In this method user labels a sample, which may be from a set of unlabeled examples or synthesized by the learning program. This method optimizes the model quality by actively acquiring knowledge from human users given a constraint on how many examples they can be asked to label [4].

In general, supervised learning includes a training set of N training examples of the form $\{(x_1, y_1), \dots, (x_N, y_N)\}$ such that x_i is the feature vector of the i -th example and y_i is its label. Object in the process is to learn a mathematical function f that can be evaluated on the input x to yield a prediction of class y . In supervised machine learning features that show little variation across samples, or are not ‘interesting’ should be filtered out. There should be also a distance, or similarity measures for two samples if they are close each other. Feature selection is important phase in supervised classification procedure. If using cross-validation in feature selection, the feature selection should be performed at each iteration.

Despite chosen classification method, classifier must be trained using the available training samples. The performance of a classifier depends on both the number of available training samples as well as the specific values of the samples. The final goal is to classify future test samples, which differ from the training samples.

Sentiment classifier can be for example SVM (Support Vector Machine), Naïve Bayes, Random Forest or k-NN (k-Nearest Neighbor). Trying different classification methods will show you which of the algorithms give the best classification result for the data. Different algorithms deliver different results and some classifiers might work better with specific feature selection configuration. It is said that state of the art classification techniques such as SVM would outperform more simple techniques such as Naïve Bayes [18] [12]. Nevertheless, it can be the opposite. Sometimes Naïve Bayes is able to provide the same or even better results than more advanced methods.

There is no single algorithm, which would perform well in all topics, domains and applications. The accuracy of some classifier can be as high as 90% in one domain/topic and as low as 60% in some other. For example for restaurant reviews Max Entropy with Chi-square as feature selection is the best combination, and for Twitter data the Binarized Naïve Bayes with Mutual Information feature selection is a good selection [18]. In Twitter data classification task, odd results can be seen and lexicon-based techniques should be avoided because of the use of idioms, jargons and Twitter slangs that affect strongly to the polarity of the tweet.

When classification predicts categorical (discrete, unordered) labels, regression model predicts continuous-valued functions. Regression is used to predict missing or unavailable numerical data values rather than discrete class labels. Regression analysis is a statistical methodology that is most often used for numeric prediction. Regression also encompasses the identification of distribution trends based on the available data [6].

In case of regression, more than two measures could be predicted from one feature. For example giving an image of a horse and wanting to know the height, weight and sex. In this case, each labeled training example is a pair of an object and the associated numerical value [2]. The quality measure of a learned prediction function is a square of the difference between the predicted value and the true value. Sometimes the absolute value of this difference is measured instead [2].

Next chapters embody algorithms for supervised classification. Prediction models are explored via literature scientific sources. Each of these algorithms has well qualitative to be chosen and tested. In Figure 3 (modified from example of [13]), the boundary differences between selected classifiers are represented for the same data. The plots show positive features as red and negative features as blue. Circles of solid colors represent training data and semi-transparent circles testing points. Number in the right lower corner represents the classification accuracy on the test set. Comparison is done on synthetic datasets [13], which makes the conveyed intuition uncertain over real dataset. When there are high-dimensional spaces involved, Naïve Bayes leads to better generalization than other classifiers.

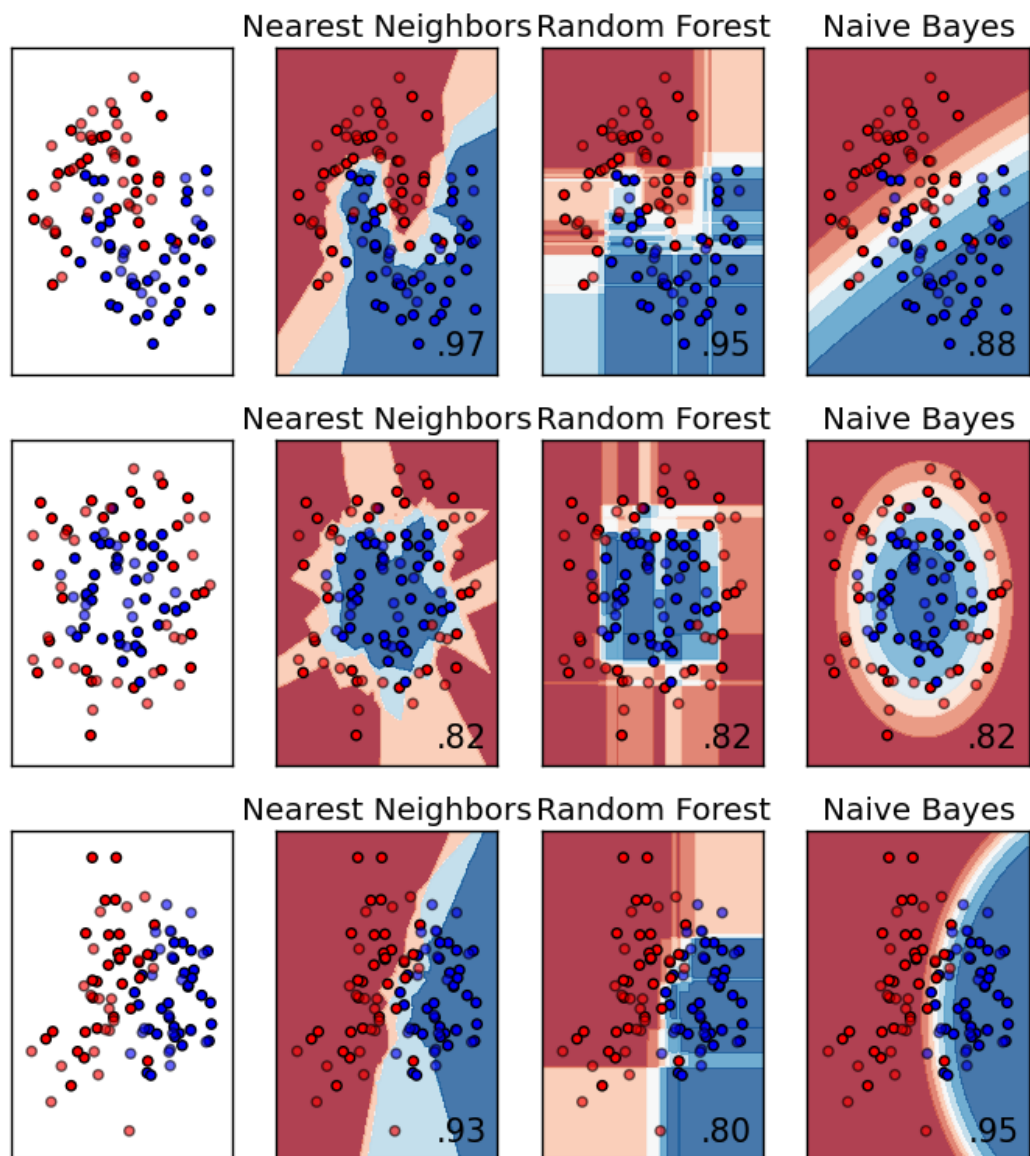


Figure 3. Decision boundaries for different classifiers.

2.3.1 Random forest

Commonly used method in data mining is decision tree learning. It is a combination of mathematical and computational techniques to aid the description, generalization and categorization of give data. The goal in decision tree learning is to create a prediction model for a target variable based on several input variables. Tree is split from the source into derived subsets based on an attribute value test, and the split is done in a recursive manner called recursive partitioning. When splitting does not add any value to the prediction or when the subset at a node has the same value of the target variable, the recursion is completed.

Random forest is a special type of classifier developed from decision tree. Decision tree is trained by an iterative selection of individual features that are most salient at each node of the tree. Tree classifier has its advantages, for example the speed and the possibility to interpret the decision rule in terms of individual features. The criteria for feature selection and tree generation include the information content, the node purity, or Fisher's criterion.

Random forest is a substantial modification of bagging combined with random selection of features. Bagging is another mechanism that uses large collection of de-correlated trees, and averages them. Difference is that Random forest uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of features. The reason for doing this is the correlation of the trees. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Significant improvements in classification accuracy have resulted from growing an ensemble of trees and letting them vote for the most popular class [1]. Random forest learning method operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. Random forest allows constructing a collection of decision trees with controlled variance.

Decision trees are a popular method for various machine learning tasks. Trees that are grown very deep tend to learn highly irregular patterns and overfit the training sets. This is caused by their property to have low bias and very high variance. [5] Historically, the bias-variance insight was borrowed from the field of regression, using squared-loss as the loss function [17]. Random forests are a way of averaging multiple decision trees, trained on different parts of the same training set, with the goal of reducing the variance [5]. Training algorithm for Random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. See algorithm 4.

For $b = 1$ to B :

1. Draw a bootstrap sample Z^* of size N from the training data.
2. Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached
3. Select m variables at random from the p variables.
4. Pick the best variable/split-point among the m .
5. Split the node into two daughter nodes.

Output the ensemble of trees $\{T_b\}_{b=1}^B$.

Algorithm 4. *Random Forest for Regression or Classification.*

To make a prediction at a new point x :

$$\text{Regression: } f_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x).$$

Classification: Let $C_b(x)$ be the class prediction of the b th random-forest tree. Then, $C_{rf}^B(x) = \text{majorityvote}\{C_b(x)\}_1^B$.

Since trees are notoriously noisy, they benefit greatly from the averaging. Moreover, since each tree generated in bagging is identically distributed (i.d.), the expectation of an average of B such trees is the same as the expectation of any one of them [20]. This means the bias of bagged trees and individual trees is the same, and improvement can be done only through variance reduction. This can be seen as opposed to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not i.d.

An average of B i.d. random variables, each with variance σ^2 , has variance σ^2/B . If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation p , the variance of the average is

$$p\sigma^2 + \frac{1-p}{B}\sigma^2.$$

The idea in random forest (Algorithm 4) is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables. Typically, values for variables m from variables p are even as low as one [20].

2.3.2 K-Nearest Neighbor

K-Nearest Neighbor (k-NN) algorithm is fundamental and simple non-parametric method for classification and regression. It is used to test a degree of similarity between documents and k training data and to store a certain amount of classification data and determine the category of the test document. The input in both cases, in classification and regression, consists of the k closest training examples in the feature space. Output in classification is a class membership where an object will be classified by a majority vote of its neighbors. This means that the object will be assigned to the class most common among its k nearest neighbors. Typically k is a user-defined constant, small positive integer.

In case of regression, the output is the property value for the object, which value forms as the average of the values of its k nearest neighbors. For both cases in machine learning, k-NN algorithm is among the simplest algorithms. Figure 4 shows example of k-NN classification. The test sample (green) should be classified either to the class of blues or reds. If $k = 3$ (solid line circle), it is assigned to class red, because there are two reds and only one blue inside the circle. If $k = 5$ (dashed line circle) it is assigned to the class blue (3 blues vs. 2 reds inside the outer circle).

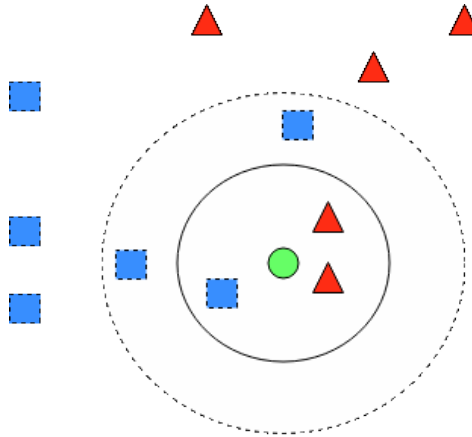


Figure 4. *Example of k -NN classification problem.*

In classification cases where there is little or no prior knowledge about the distribution of the data, k -NN is a good choice to study. K -NN classification was developed for discriminant analysis and is a type of instance-based learning algorithm, where the goal is to categorize the objects based on closest feature space in the training set. Here function is only approximated locally and all computation is deferred until classification. Because k -NN uses only the training point closest to the query point, the bias of the 1-nearest-neighbor estimate is low, but the variance is high. Asymptotically the error rate of the 1-nearest-neighbor classifier is never more than twice the Bayes rate [7].

Training examples in k -NN are vectors in a multidimensional feature space, each labeled with a class. The training phase of the algorithm only stores the feature vectors and class labels of the training samples. Distance between vectors is typically computed with Euclidean Distance. This method provides availability of a similarity measure for identifying neighbors of a particular document. In the classification phase an unlabeled vector is classified by assigning the label, which is most frequent within query point. The best choice of k depends on data. Noise of the classification reduces if k gets large values, but in this case, boundaries between classes are less distinct.

The accuracy of the k -NN algorithm depends on the presence of noisy or irrelevant features. In two class (binary) classification, it is helpful to choose k to be an odd number as this avoids tied votes. Validation of results of a k -NN classification is often done with a confusion matrix. Confusion matrix is a specific table layout that allows visualization of the (typically a supervised learning) performance of an algorithm.

2.3.3 Naïve Bayes

Naive Bayes (NB) classifier is a simple probabilistic classifier based on Bayes theorem. It is a popular machine learning algorithm for text classification, and it outperforms alternatives that are far more sophisticated. Algorithm applies “naive” assumption of independence between every pair of features. This means that the presence or absence of a

particular feature of a class is unrelated to the presence or absence of any other feature. Each feature contributes independently to the decision of which label should be used. This can be seen problematic when more than two of the features are correlated with another.

Another concern is that the individual class density estimates may be biased, but it does not hurt the posterior probabilities much, when occurring near the decision regions. During its operation, naive Bayes assumes a stochastic model of document generation. Using Bayes' rule, the model is inverted in order to predict the most likely class for a new document. In spite of apparently over-simplified assumptions, naive Bayes classifiers have worked well in document classification for example in spam filtering. NB requires a small amount of training data to estimate the necessary parameters. In addition, naive Bayes learners and classifiers can be extremely fast compared more sophisticated methods.

A classifier based on naive Bayes algorithm [10]:

In order to find the probability for a label, this algorithm first uses the Bayes rule to express $P(\text{label} | \text{features})$ in terms of $P(\text{label})$ and $P(\text{features}|\text{label})$:

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(\text{features}|\text{label})}{P(\text{features})}.$$

The algorithm then makes the 'naive' assumption that all features are independent, given the label:

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(f_1|\text{label}) * \dots * P(f_n|\text{label})}{P(\text{features})}.$$

Rather than computing $P(\text{features})$ explicitly, the algorithm just calculates the denominator for each label, and normalizes them so they sum to one:

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(f_1|\text{label}) * \dots * P(f_n|\text{label})}{\sum [I] (P(I) * P(f_1|I) * \dots * P(f_n|I))}.$$

Two probability distributions parameterize the classifier. 1) $P(\text{label})$ gives the probability that an input will receive each label, given no information about the input feature. 2) $P(f_{\text{name}}) = P(f_{\text{val}}|\text{label})$ gives the probability that a given feature (f_{name}) will receive a given value (f_{val}), given that the label (label). If the classifier encounters an input with a feature that has never been seen with any label, then rather than assigning a probability of zero to all labels, it will ignore that feature [2].

There are 3 types of Naive Bayes classifiers. These differ mainly by the assumptions they make regarding the distribution of $P(\text{feature}|\text{label})$. Gaussian naive Bayes implements the Gaussian naive Bayes for classification where the likelihood of the feature is assumed Gaussian. Multinomial naive Bayes implements algorithm for multinomially

distributed data, and is one of the two classic naive Bayes variants used in text classification, where the data is represented as word vector counts.

Multinomial NB classifier is suitable for classification with discrete features and normally multinomial distribution requires integer feature counts. However, also fractional counts like TF-IDF vectors are known to work well in practice. In case of TF-IDF, the distribution is parameterized by vectors for each class, where the number of features is the size of the vocabulary (in text classification).

Bernoulli Naive Bayes is for training classification algorithms for data that is distributed according to multivariate Bernoulli distributions. For example if there are multiple features but each of them is assumed a binary-valued variable. This class requires samples to be represented as binary-valued feature vectors.

The decision rule for Bernoulli naive Bayes is based on:

$$P(xi|y) = P(i|y)xi + (1 - P(i|y))(1 - xi).$$

This differs from multinomial naive Bayes rule in that it explicitly penalizes the nonoccurrence of a feature i that is an indicator for class y , where the multinomial variant would simply ignore a non-occurring feature. In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. Bernoulli NB might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits. Bernoulli model is particularly sensitive to noise features and requires some form of feature selection or else its accuracy will be low.

2.3.4 Comparison between above algorithms

The comparison of the classifiers and using the most predictive classifier is very important. Based on datasets each methods show different efficacy and accuracy. The goal is to find the most accurate classifier for text classification task where the classification subject can change. For example, when classifying sentiments from the tweets, we can search different kind of feelings. Whether it is just 'yes' or 'no' to something, or we are searching content that makes people cry.

There is a relationship between Random forests and the k-Nearest Neighbor algorithm. Both of these algorithms can be viewed as weighted neighborhoods schemes. When comparing decision trees, which a Random forest is, and Naïve Bayes, the result is more complicated. Decision trees are very flexible, easy to understand and debug, and they will work in classification and regression problems. Decision trees will handle predicting a categorical value like (red, green, up, down) and continuous value like 1.5, 4.2. Decision trees need only a table of data and they will build a classifier directly from that data without needing any up front design work to take place. Because Random forest is

more advanced than simple decision tree, it does not tend to over fit the training data. To get the best performance out of decision trees, Random forest perform quite well. This adds options to tune and more implementing.

Naïve Bayes classification must be built by hand. When decisions trees will pick the best features from tabular data, picking features for Naïve Bayes is up to user. Bayes can perform well and it does not over fit like decision trees. This means simpler algorithms to implement. However, Naïve Bayes is harder to debug and understand because it is all probabilities multiplication. When the training data contains certain possibilities, Naïve Bayes works quite well. When there is a lot of data, decision trees work better compared to Naïve Bayes.

Decision trees are handy because they tell what inputs are the best predictors of the outputs. If there is a statistical relationship between input and output, using decision trees will show how strong that relationship is. Best way to research the best classifier is to test and compare the results.

2.4 Validation / Evaluation

When trying different training data, feature selection methods and classification algorithms, validating the results is the thing that shows how the combination of these phases works together. Measurement for this is the accuracy of each classifier. Accuracy measure is not a reliable metric for the real performance of a classifier. This is caused if there is a different amount of data in each class. If taking randomly 90% of the data for learning and 10% testing, classifier is easily biased into classifying to the class, which is the most present in the learning data. To get good validation metric for the classifiers other techniques must be involved and these are the cross-validation and confusion matrix.

Cross-validation is a model validation technique to test how the results of a statistical analysis will generalize to an independent data set. Cross-validation is used when estimating the accuracy of a predictive model. Model is a dataset of known data (training) and unknown data (test). Goal is to define a dataset to test the model in the training phase, and limit problems like over fitting or get insight how the model generalize to an independent dataset. Cross-validation takes time, because each round of cross-validation partitions a sample of data into subsets and performing the analysis on one subset (training set), and then validating the analysis on the other subset (test set). Multiple rounds of cross-validations are performed using different partitions to reduce variability, and finally all the validation results are averaged to get results. There are two types of cross-validation, exhaustive and non-exhaustive cross-validation. Exhaustive cross-validation learns and tests all possible ways to divide the original sample into training and a validation set. Examples of these are Leave-p-out cross-validation and Leave-one-out cross-validation. Non-exhaustive cross-validation methods do not compute all ways of split-

ting the original sample. Example of non-exhaustive cross-validation is k-fold cross-validation and repeated random sub-sampling validation.

Another testing measurement is to run confusion matrix for classifiers. Confusion matrix is also known as contingency table or an error matrix. This specific table allows visualizing the performance of an algorithm and it is used typically in supervised learning. Here each column of the matrix represents the instances in a predicted class, and each row represents the instances in an actual class. Table shows easily if the system confuses classes. For example if wanting to distinguish between positive, neutral and negative, a confusion matrix will summarize the results of testing the algorithm for further inspection. Table 1 shows resulting confusion matrix for an example where total amount of samples (30 tweets), includes 12 positive, 8 neutral and 10 negative.

Table 1. *Example of confusion matrix.*

		Predicted class		
		Positive	Neutral	Negative
Actual class	Positive	10	2	0
	Neutral	3	4	1
	Negative	0	2	8

In this confusion matrix, system predicted that two of twelve positives were neutrals, and from the neutrals (8), system predicted that 3 were positives and 1 negative. From eight negatives system predicted that two of them were neutrals. Matrix shows, that the system has trouble distinguishing between neutral and positives, but can make the distinction between negative and positive well. Correct guesses are in the diagonal of the table and makes inspecting the results visually easy.

3. IMPLEMENTATIONS

This section of the thesis contains the implementation phases that were used and tested to make most accurate sentiment analyser. Building the analyser started with a lexicon based sentiment analysis but the approach were changed later to learning based methods. Analyser was built in flexible way, so it can be used even though the subject of analysis changes. This makes modeling the training data more difficult than using topic specific training data.

Implementations chapter describes the classification process and different techniques that were used. Chapter starts with an introduction to the process as a whole and moves to explain in more detail data retrieval, feature extraction, feature selection and classification. For each of the previous mentioned sections few different methods were tried and the best performing combination was selected.

Process started with studying the structure of training and test data. How to get data from Twitter and what pretreatment does the data need. The accuracy of different classifiers varied during testing and implementing different feature extraction and selection methods. The goal was to build as accurate classifier as possible to classify sentiments out of Finnish tweets. Fine-tuning the analyzer continued after the SuomiLOVE project by modifying training data and trying out different feature selection methods.

Figure 5 shows different parts of the analyser and chapter goes each phase through with some examples. Chapter 3.1 starts describing the process from the data retrieval from Twitter. Twitter was sensible choice as a social media platform due to its diverse application-programming interface for developers and amount of available data. Chapter encompasses manual human rating of training data and how the data is handled from and to the database. After the training data is labeled into correct sentiment classes, classification process can begin. Classification process starts with feature extraction and moves through feature selection to the classification. This work comprises TF-IDF feature extraction, few feature selection methods and three classification algorithms.

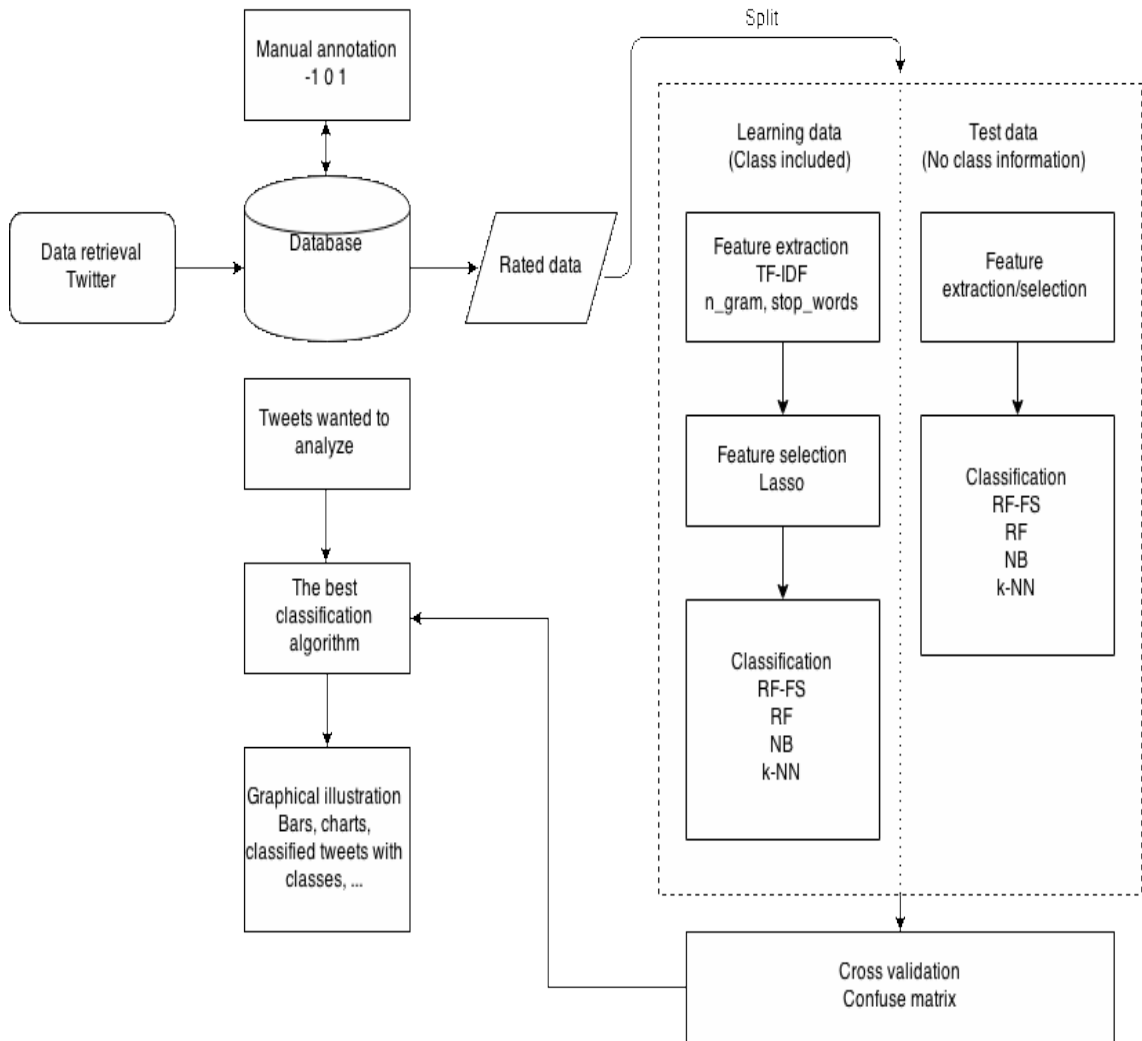


Figure 5. Block diagram of the implemented analyser.

3.1 Data retrieval

Getting data out of Twitter is maybe the fastest and easiest selection for the first timer when observing differences between social media platforms and how to get data out of them. Twitter has a great Application Programming Interface (API) for developers and one tweet contains a lot of information. Tweet is a short message sent using Twitter, which can include text and/or media. One tweet can be maximum 140 letters long, but it includes a lot more metadata. Tweet is usually a bundled text with two additional pieces of metadata, which are entities and places. Entities are users (@username), hashtags (#thesis), URLs (http://...) and media that may be associated with a tweet. Places are locations in the real world like school, cafeteria, city or country.

Twitter metadata includes a lot of information about the user and this enlarges the possibilities with analyzing data from different perspectives. Table 2 shows what information includes in one tweet.

Table 2. *Metadata of one tweet.*

<tweet's unique ID>
<text of the tweet>
<when the tweet is created>
<the ID of an existing tweet that the tweet is in reply to>
<user screen name>
<user ID of replied to tweet author>
<the authors user ID>
<authors biography>
<authors screen and user name>
<authors URL>
<authors location>
<rendering information for the author>
<creation date for the account>
<is the account contributors enabled>
<number of favorites which user has>
<the author of the tweet>
<number of tweets that user has>
<number of users that the user is following>
<time zone and offset>
<selected language>
<is the user protected or not>
<number of followers>
<does user has a verified badge>
<the place ID>
<geo tag (if enabled)>
<printable names of the place>
<contributors ID>
<URL to fetch a detailed polygon for the place>
<type of the place>
<place associated with the tweet>
<country where place is>
<application that sent the tweet>
<bounding box for the place>

To access Twitter API, a Twitter account is required for creating an application for developers. This standard allows Twitter to monitor and interact with third party platform developers as needed. With an authorized API connection, a request can be issued. API is crafted intuitive and it is easy to use. Available libraries make usability even easier. For example, in Python there is an available package called *twitter*.

The purpose of using Twitter API is to pick tweets from the stream, where a specific hashtag or word is used. A Representational State Transfer (REST) API together with open standard for authorization (OAuth) allows programmatically reading and writing Twitter data. Because of the nature of the analyser, Streaming APIs is needed, which continuously deliver new responses to REST API queries. Responses from REST API are available in JavaScript Object Notation (JSON) serialization format.

At the beginning, receiving tweets were done manually from old tweets. This was due to the reason that we used the REST API algorithm and not the Streaming APIs. About 2000 tweets were needed from the analyzing subject that the classifier could be taught properly. It might be expected that Finns don't tweet much because we don't speak

much, but that is not correct. Collecting tweets happened fast and Twitter API helped in a way, that it enables to search at least two weeks old tweets. This is a profitable feature also when streaming encounters errors.

After receiving the tweets through Twitter API, tweets are stored to a database for later use. Database runs important part of the work and has several different sections for the data, each of them having a specific meaning. Database collects 1) requested tweets, 2) manually classified tweets and 3) classified tweets by the machine. Incoming tweet contains in excess of 5 KB of total content when represented in uncompressed JSON [14]. This means more than 40 times the amount of data that makes up the 140 characters tweet.

Data that were mined during SuomiLOVE included hashtags *#suomilove* and *#suomiloveyle*. These hashtags identified the tweets for the TV-show. To teach the machine, training data before the show begins were needed to collect. Training data were collected by querying emotional tweets in Finnish, including love, affection and other strong sentimental tweets.

For training, equal amount of labeled tweets from each of the classes are needed. Training data needs manual annotating and by using, a web-based application where user classifies the tweets from the stream can speeds the process. When person has classified apparent tweet, application shows the next one to be classified. All retweets are deserted from the rating process to avoid duplicates.

When starting to classify tweets, in addition to positive/neutral/negative there can be classes for other sentiments. Tweet belongs always into one of the previous 3 classes but it might be classified also into another class, representing more specific a certain sentiment. In this work, more specific sentiment classes were happiness, sadness, joy, hate, sarcasm and some others, but the final system were simplified and uses only positive/neutral/negative classes. Table 3 shows the number of tweets for each class and notes what kind of sentiments or type of tweets are classified into each one of them.

Table 3. *Division into three classes.*

Class	Number	Notes
Positive	700	happy, supportive, loving
Neutral	700	statement/observation without clear sentiment
Negative	700	disappointment, hate, dislike

All tweets in training data need a label (class). Manual labeling to different classes is

time consuming, but obligatory if wanting to get correct labels for teaching the machine. When making sentiment analysator for SuomiLOVE, all tweets that were happy, loving and supportive were labeled as a positive. There were almost none of negative tweets, so classifying into three classes did not quite work in that subject. For this reason, mission changed to only analyzing the positive tweets out of the stream. Later when starting to develop the algorithm for another project, all kinds of tweets were labeled to get standard language sentimental tweet data. 2100 sentimental tweets were labeled equally distributed as positive, neutral and negative from different subject categories.

3.2 Lexicon based sentiment classification

Sentiment analysis project started with a lexical approach. The aim was to use learning based sentiment classification, but lexicon approach was fast way to try what is the main point behind sentiment analysis. Several previous studies can be found from the Internet, which use lexical approach to extract the sentiments out of tweets. These studies [8] [16] were used as a base for the first version of the sentiment classifier.

Because lexical approach uses dictionaries and the method counts the sentiment level out of tweets with integers, using an English vocabulary from same kind of sentiment analysis project was used. Vocabulary was translated it into Finnish and the list was inspected so that the values from 5 to -5 for each word add up. This means, that every word in the list gets a value from range 5 to -5, where 5 means extremely positive and -5 extremely negative. Scores for words were given and inspected manually by 4 person.

For example a sentence '*Minä rakastan sinua*' (I love you), get values; *mina*(0), *rakastan*(5), *sinua*(0) and that makes as a total 5, and is extremely positive sentence. Another example could be '*Vihaan tätä laulua, mutta artisti on super ihana*' (I hate this song, but the artist super adorable). Words in this sentence get values; *Vihaan*(-5), *tätä*(0), *laulua*(0), *mutta*(0), *artisti*(0), *on*(0), *super*(3), *ihana*(4). This sentence gets total of 2 points and is classified slightly positive, albeit it also includes a negative opinion.

Formulas were implemented with Python to calculate a total value for one tweet, which arrived in real time from Twitter API. Polarity of the tweet was calculated from the word values and they were presented in a bar chart at a webpage. Lexical analysis was tested for live football match between Finland and Hungary. It seemed to work nicely, but there was clearly noticeable effect, that almost every other tweet went to neutral class, keeping positive and negative bars low. This happened because the tweets contained many sport specific words that were not in our vocabulary.

Lexical approach was effective demonstration about the idea behind the sentiment analysis. It is said that lexical approach is no good for Twitter data, because of the slang that people use there. For this reason learning based method for sentiment classification was a better choice.

3.3 Learning based sentiment classification

Interest towards machine learning and using much-vaunted Bayesian formulas to extract sentiments from tweets brought the second choice for sentiment analysis; the learning based technique. Original idea was to use Naïve Bayes as a classifier. More classifiers came along during the implementation process. Some of methods did not work for Finnish Twitter data and finally 3 classifiers were chosen for our sentiment analysis. These 3 classifiers based on supervised learning are Naïve Bayes, k-Nearest Neighbor and Random forest.

Python was used as a main programming language, because it has excellent functionality for processing linguistic data. It is also free, simple and powerful programming language, that has a shallow learning curve, good string-handling functionality and its syntax and semantics are transparent. As a Python IDE (Integrated Development Environment), we used Pycharm. Pycharm offers first-class support and advantages to use extensive standard libraries, including components for graphical programming, numeric processing and web connectivity.

The implementation process was done with Python using mainly scikit-learn and NLTK packages together with several free software packages. NLTK is a leading platform for building Python programs to work with human language data. Scikit-learn is a simple and efficient tool for data mining and data analysis. Scikit-learn contains all needed classes to perform text data analysis. Other packages used in the process are presented in Table 4.

Table 4. *Python packages for the project.*

Name	Description
NumPy	Package provides substantial support for numerical processing in Python. Numpy has a multidimensional array object, which is easy to initialize and access.
Pandas	Package provides high-performance, easy-to-use data structures and data analysis tools for Python.
Matplotlib	Package supports sophisticated plotting functions with a MATLAB-style interface.
CSV	Package to read and write files stored in comma-separated-values file format.

3.4 Feature extraction

Feature extraction is the part where features are extracted from the labeled tweets from the database. These extracted features are for training and testing the classifier. In addition to feature extraction algorithms together with Python, scikit-learn and mixing it with NLTK needed more examination. Combination of these two enables creating working environment for human language analysis. The condensed idea behind feature extraction is to separate words out of sentences and tag them with counts or TF-IDF's.

To extract features from labeled tweets, training data must be converted from JSON into CSV format. After the converted data is loaded, it is separated, such that 90% is used for training and 10% for testing. The samples are selected randomly for training and testing at each time and this can cause deviation to accuracy.

Data is separated equally from each class, that training data contains the same amount of tweets for positive, negative and neutral classes. The split is done in a stratified manner, which means dividing the train and test indices into train and test sets. Cross-validation object in this is a variation of k-fold that returns stratified folds. The folds are made by preserving the same percentage of samples for each class. The goal of the cross-validation is to estimate the expected level of fit of a model to a data set that is independent of the data that were used to train the model [3].

Feature extraction can be done in two possible vectorizer functions; a count vectorizer and TF-IDF vectorizer. Common usage for vectorizer is a count vectorizer, which converts a collection of text documents to a matrix of token counts. This method implements both tokenization and occurrence counting in a single class. The implementation produces a sparse representation of the counts using sparse matrices. Count vectorizer can have many parameters, but the default values are usually reasonable for feature extraction. The number of features will be equal to the vocabulary size if any a-priori dictionary is not provided.

Another vectorizer function, TF-IDF vectorizer converts a collection of raw documents to a matrix of TF-IDF features. TF-IDF feature extraction can be done also combining all the options of count vectorizer and TF-IDF transformer in a single model. TF-IDF transformer transforms a count matrix to a normalized TF or TF-IDF representation. TF-IDF vectorizer defines each sentence as a vector and in each vector, the numbers (weights) represent features TF-IDF score [13]. This work uses TF-IDF vectorizer as a feature extractor.

When extracting data, vectorizer functions offer a choice to use n-grams. N-gram is a contiguous sequence of n items from a given sequence of text. This is given as a ple (min_n, max_n), where min and max values are the lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that

$min_n \leq n \leq max_n$ will be used [13]. When using n-grams, the classifier ‘understands’ more. Testing n-gram ranges (1), (1, 2) and (1, 3) indicated small differences. Difference between these methods starts to increase if using higher values, for example (2, 3). Because of the structure of Finnish sentences, option for n-grams (1, 3) was the best choice.

Another useful characteristic in vectorizer function is a *stop word list*. Stop words means words that do not make any difference in a sentence when extracting sentiments and are filtered out before or after processing text data. If choosing to use stop words they are usually set as a string or a list. Ready-made stop word list strings can be found for some languages. This work uses a custom-made stop words list. List consisted from 200 most frequent words, which were present in the training data. These 200 were chosen, because most likely they don’t include any sentimental information and this is quick way to test the list.

From that 200-word list, all numbers, nouns and adjectives relative to the analyzing subject were deleted. All Finnish personal pronouns, prepositions and conjunctions were added to the list. This operation took some words away from the list and the final list included 175 words. When testing how the classifier reacts on stop words list, it can be seen, that some words cannot be in the list. Reason for this is that those words affect the meaning of some another word too much. For example, words ‘*kyllä*’ (yes) and ‘*ei*’ (no) were removed from the list. This is somewhat problematic, because we don’t want to automatically classify either of these words to one of the classes.

One text-preprocessing task for feature extraction would be stemming. In other words, lemmatization is dependent on the language of the text. Purpose of lemmatization is to reduce a word to its dominant mode, so that similarity detection can be achieved. When using word stems to improve the accuracy of the analyser, a simple approach would be stripping off anything that looks like a Finnish suffix. Finnish list contains 86 suffixes, but when stemming in Finnish, some words are not the same or recognizable anymore. It is argued, that applying lemmatization techniques to a piece of text may affect the semantics.

Another aspect for preprocessing is the spelling mistakes, replacing acronyms and abbreviations. As we see, people misspell quite often and when they are tweeting in a hurry, it is easy to make a mistake. If correcting spelling mistakes from the data, the process would eliminate noisy data before the main process starts.

3.5 Feature selection

Feature selection chooses the most relevant features from the data and is one of the most important phases in classification. Feature selection module in classification process can be used for selecting features or reduce dimensionality on sample sets. These operations

can be used either to improve estimators' accuracy scores or to boost performance on very high-dimensional datasets.

There are several competent techniques for feature selection. For the text data, variance threshold, univariate feature selection, forward selection, recursive feature elimination, L1-based feature selection and feature selection as part of a pipeline can be tried as a feature selection method.

Variance threshold is a feature selector that removes all low-variance features [13]. This method is better to be used for unsupervised learning because it looks only at the features, not the desired outputs. The second tested feature selection method was univariate feature selection. This selection method works by selecting the best features based on univariate statistical tests [13]. For this selection, for example Chi-square and Select K-Best that implement the transform method can be used.

Chi-square computes stats between each non-negative feature and class. This score is used to select the n features with the highest values from X , which contains only non-negative features such as frequencies (term counts in document classification). These frequencies are relative to the classes. Chi-square test measures dependence between stochastic variables. In another words, this function removes the features that are most likely to be independent of class and is therefore irrelevant for classification.

Select K-Best selects features according to the k highest scores. Select K-best includes two parameters; score function and k . Score function takes two arrays X and y , and returns a pair of arrays indicating *scores* and *pvalues* (significance). The number of k , is the number of top features to select. How the score is calculated depends on the used filters. Some of the choices are represented in Table 5. Ties between features with equal scores will be broken in an unspecified way [13].

Table 5. Score functions for Select k -best feature selection algorithm.

Name	Description
Chi-square	Chi-squared stats of non-negative features for classification tasks.
Select Percentile	Select features based on percentile of the highest scores.
Select False positive rate	Select features based on a false positive rate test.
Select False discovery rate	Select features based on an estimated false discovery rate.
Generic Univariate Select	Univariate feature selector with configurable mode.

Backward selection, or in other words, Recursive feature elimination (RFE) ranks features given an external estimator that assigns weights to features. Parameters for estimator are the objects, which in supervised learning is a fit method that updates a coefficient attribute that holds the fitted parameters. Important features must correspond to high absolute values in the coefficient array [13]. Another parameter that RFE includes is the number of n feature to select. If using none as a default, half of the features are selected. RFE includes also a step (default=1), which corresponds to the integer number of features to remove at each iteration.

L1-based feature selection is also called a Lasso. It is a linear model, which is trained with L1 prior as regularizer and estimates sparse coefficients. Lasso reduces effectively the number of variables and suits well for feature selection. Lasso has a constant alpha that multiplies the L1 term and is a good at finding useful features when there are many of varying quality.

Feature selection is usually a part of pre-processing step before doing the actual learning. Recommended way to do feature selection is to use *Pipeline* in classification. In this snippet, features importance is evaluated and the most relevant features are selected. Then training a classifier, for example Random forest on the transformed output using only relevant features. Pipeline can be used to similar operations with the other feature selection methods and for classifiers that provide a way to evaluate feature importance.

The Pipeline is built using a list of pairs (*key*, *value*). Here the key indicates a string containing the name of the step and the value is an estimator object. Making a pipeline

can be done shorthand, using utility functions, which takes a variable number of estimators and returns a pipeline, filling in the names automatically.

3.6 Classification

Choosing the right classifier and make the classification for extracted and selected text features is the last part of the training process. As previous studies have indicated, there are many possible choices for text classifiers and the selection depends entirely on the data that needs to be analyzed. When the data is from social media, where people use many smileys, abbreviations and slang, text differs from for example books reviews. Text includes more typos and things which true meaning some persons can't even guess. Example of this kind of saying is '*yolo*' used by teenagers or abbreviation of some TV-show like '*TVOF*' (The Voice of Finland).

Classification in this work is done into three different classes; positive, neutral and negative. In addition to these basic sentiment categories, there were classes for more specific feelings like happiness, laughter, cry/tears, sarcasm and hate. Classifying for smaller groups remained in the background, but ambition is that someday the text can be categorized into more accurate sentiment categories.

At this stage, feature vectors including every feature that the feature extraction and selection process did not erase are defined. When passing the feature vector for the classifier, classifier will take it and learn to recognize the important tweets and their polarity. If word '*kaunis*' (beautiful) is present usually in positive class, more frequently than in negative class, the classifier learns to recognize this word and make assumption, that the tweet containing this word goes to positive class.

Classifier counts the frequencies of each feature in the feature vector and their classes and makes assumptions. If some important word is taught equally for two different classes, classifier seeks what are the words around this important word. In Finnish, '*kaunis*' is a positive word, but if there is the word '*ei*' (no) or words '*ei ole*' (is not) meaning changes and tweet should be classified into class negative.

After the classifier is taught with the training data, it is tested with the test data set. When the classifier has reached a wanted accuracy level, it is used for unseen tweets from the data stream. Classifier will take one tweet at a time into examination and checks the matching label (class) for the tweet. 10 percent of total data is used as a test data, but in real action, new real time arriving tweets goes into classification. Tweets are collected with specified qualities and purpose of the classifier is to recognize from the features, in which class certain tweet belongs.

Naturally, when making an analyser it is important to see how the classifier manages to make the classification process. By computing each classifier's accuracy level for the

test set, the best performing classifier can be found. Testing is done for the test set which is divided from the total amount of labeled data. Classifier classifies the test set and accuracy is counted for the test set based on labels that the test set has. Accuracy measure shows how many test tweets were labeled correctly in the classification process for test data. Checking manually how the classifier labeled test tweets gives a nudge in the right direction how to modify data further.

When accuracy is between 0.70 and 0.80, machine can do quite good labeling and the importance and need of human labeling emphasizes. Human concordance is threat to accuracy in sentiment analysis, because humans don't agree universally with one another. Getting 1.0 (100%) accuracy with only help of a machine is hard or not possible for large and various text data like from Twitter. This happens because some sentences are not easy to analyze and manual annotating can be ambiguity.

When classifying into 3 categories, positive, neutral, and negative, the data consists with tweets that are objective and subjective. While usually subjective tweets express some sentiments, objective does not. This is in some cases not that simple. For example, *"I think Finland is a hockey country"* is a subjective tweet and goes to neutral class, because it does not include any sentiment. However, a tweet *"Years 1995, 2011 Finland won gold, but usually they keep losing to Sweden"*, is an objective as it states a fact, but it expresses an implicit opinion: *"losing to Sweden"* which is more to be negative. A Finnish person knows that the last tweet has a negative sentiment, but how would the machine know that losing to Sweden is not a good thing.

There are also other factors influencing the accuracy. These five factors are shown in Table 6.

Table 6. *Main factors influencing the accuracy of text classification.*

Factor	Description
Context	Positive and negative sentiment word can have the opposite connotation depending on context. “ <i>You’re great!</i> ” is a positive while “ <i>Great job by blowing my day!</i> ” is negative.
Sentiment ambiguity	Tweet can be neutral even it has a positive or negative word. For example “ <i>Can you recommend any fantastic movie?</i> ” does not express any sentiment, but it does use the positive sentiment word “ <i>fantastic</i> ”. In addition, a tweet “ <i>This DVD-player is slow</i> ” is negative even though it does not express any sentiment word.
Sarcasm	If there is sarcasm involved in a tweet, positive can be quickly negative. For example, “ <i>I’m so glad, that I drop my wallet for that thief to be found</i> ” is clearly sarcastic and negative; even it has the word “ <i>glad</i> ”. Sarcasm can be detected mainly from the context (losing a wallet is not a good thing).
Comparatives	If comparing something there is always two ways to see the statement. For example, “ <i>Tappara is way better than Ilves</i> ” is positive for Tappara and negative for Ilves (Finnish hockey leagues). There is a positive keyword “ <i>way better</i> ” but no regarding for whom.
Regional variations	Depending on language, a word can change sentiment and meaning. This can be seen in slang and language variations. For example, in Finnish “ <i>itikka</i> ” means <i>cow</i> and <i>midge</i> , depending on a province that you use the word.

After the classification, it is interesting to see the most informative features that were used in each class. The most informative features can be seen after mapping feature names to the feature vectors. In this case, more than two classes are needed, where m is the total number of features and n is the number of classes. If wanting to see the top 10 most discriminative features of class 1, sort the corresponding row and extract the feature names of top 10 features. Then iterate over all the indices and get the corresponding feature names. Sorting works only if the features are being normalized. Inspection of the results shows if some words are going badly into wrong class. The most informative features should be seen with common sense that is the class right.

In order to accommodate features that depend on a word's context, the pattern that was used to define the feature extractor must be revised. Instead of just passing in word to be tagged, a complete, untagged, sentence, along with the index of the target word is passed. These target words in this work are tagged with TF-IDF technique.

From all the possible text data classifiers 3 were chosen. Target was to find the most accurate classification method. Next will be explained the main reasons why this work includes Random Forests, k-NN and Naive Bayes as a classification algorithms. Results and comparison between classifiers is represented in chapter 4.

3.6.1 Random forest

Random forest was not the first selection as a classifier and is not mentioned in many previous studies in relation to text based sentiment analysis. Implementation was easy with scikit-learn and quite good results were achieved after the first classification test. Random forest did not succeed to same level as Naive Bayes at first but was close. Why Random forest is included as one of the classifiers is explained next.

Random forest can rank the importance of variables in case of classification or regression. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over fitting [13]. When measuring the variable importance, a Random forest needs to be fitted to the data. Data is fit with either two sparse or dense arrays X and y . Array X is for the training samples and array y for the target values (class labels) for the training samples.

Fitting process in Random forest includes recording and averaging out-of-bag error for each data point. The importance of the feature can be measured by permuting the values of the feature among the training data and calculate the out-of-bag error again on perturbed data set. Importance for the feature is the average of the difference in out-of-bag error before and after the permutation over all trees. Normalization is done by the standard deviation of the differences.

Random forest ranks features, which produce large values more important than features, which produce small values. Figure 6 shows the importance of features when using Random forest classifier. The red bars are the feature importance's of the forest. Plot shows that the features 7617 and 3188 are the most important following few other important features and then the importance starts to decrease for the features. Features in bars are words of n-grams (1, 3) that describe connection to a certain class. For example, course words are important negative features and celebrative and warm words describe the importance of positive class.

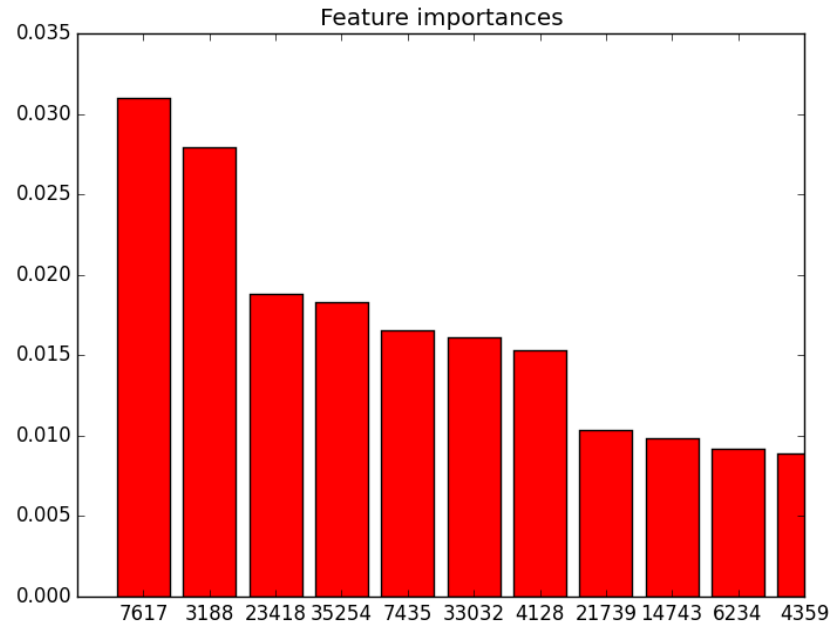


Figure 6. *Feature importances.*

Accuracy of the Random forest increases when adding right feature selection method to it. Feature selection works as a part of the model construction process. One way to use feature selection is to make a pair of Random forest and feature selector in a pipeline. For example, when using Lasso as a feature selection method, Lasso constructs a linear model from the extracted features and shrinks coefficients (features) to zero. All features having non-zero coefficient will be selected by the Lasso algorithm. One thing is to find a good alpha for Lasso, so it will fully recover the exact set of non-zero variables using only few observations.

3.6.2 K-Nearest Neighbor

K-NN was chosen because of its easiness to implement. K-NN is also in some text classification cases very effective and has good qualities because of its non-parametric form. K-NN calculates similarity between test document and each neighbor, and assigns test document to the class, which contains most of the neighbors. The category for the feature is predicted based on the nearest point, which has been assigned to a particular category. The k-Nearest Neighbor classification method is widely used because of its simplicity and is suitable technique for text classification. This method performs well even in handling the classification tasks with multi-categorized documents.

K-NN classification has some drawbacks. For example, it uses all features in distance computation, and makes the method computationally intensive. Computational intensity increases when the size of training set grows and in this research had over 10 000 fea-

tures after feature selection. Accuracy of k-Nearest Neighbor classification is degraded hard if there are noisy or irrelevant features involved.

3.6.3 Naive Bayes

Naive Bayes classifier can be trained efficiently by requiring a relatively small amount of training data to estimate the parameters necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix. An advantage of the naïve Bayes classifier is that it requires a small amount of training data to estimate the parameters necessary for classification. Bayesian classification approach arrives at the correct classification as long as the correct category is more probable than the others are. Category's probabilities do not have to be estimated very well. In other words, the overall classifier is robust enough to ignore serious deficiencies in its underlying naïve probability model.

Naïve Bayes was a certain choice to be involved in the project as a classifier, because it has been one of the popular machine learning methods for many years. The framework in Naïve Bays is attractive since its simplicity in various tasks and it offers reasonable performance, obtained in the tasks even though the learning is based on an unrealistic independence assumption.

4. RESULTS

This section summarizes the research results and their relevance. Chapter includes the most important results, sources of error and deviations from the expected results and discusses the reliability of the research. Results relate to implementation process of the work using lexical and learning based sentiment analysis for text data.

Preprocessing in lexicon based approach included forming a vocabulary and rate words that have a sentimental meaning. For example, words *love*, *hate*, *fantastic*, and *horrifying* were rated with values from -5 (extremely negative) to 5 (extremely positive). Ready-made vocabulary were found and translated into Finnish from previous (English language) study. All features and rates were inspected manually before using them for analysator. Calculating total value for one tweet was done summing the rates. Lexical analysator managed quite well, but slang, sarcasm and minimalistic vocabulary messed the classification. This resulted as high amount of neutrals and low amount of positive and negative tweets.

Data retrieval from Twitter is made easy for developers. Twitter's application programming interface enabled searching 2 weeks old tweets, which helped collecting data for training. When collecting data, search was performed using descriptive search words to collect versatile and valid training data for the machine. Content is important because the features that tweet encompass, affects the learning process and further analyses when using new tweets. Twitter's Streaming API enables real-time analyzing when the REST API together with OAuth provided collecting already existing tweets. Table 7 shows the amount of training data collected from Twitter.

Table 7. *Amount of labeled training data from Twitter.*

Class	Rate	Amount
Negative	-1	700
Neutral	0	700
Positive	1	700
Total		2100

In learning based approach, manually annotated tweet data was split into training and testing data. Split is mandatory to test the prediction of an estimator with different data than what was used to fit the estimator. When the split is done, it is important that equal amount of data is used for each class. Equality is important, because classifier is easily biased into classifying to the class, which is the most present in the learning data. Split in this study was done relative to 90-10, which means that 90 percent went for training and 10 percent for testing.

Sentiment classification was done to classes positive, neutral and negative (Table 7) where the manual annotating was done by the intuition about the sentiment that the tweet contains. This was not quite unambiguous, because one might think that the tweet is positive and some other that it is negative. Tweet that was hard to classify what sentiment it includes was labeled as neutral. Labeling was done giving values -1, 0, and 1 for every tweet using custom made rating web-application. Training data was inspected before training, so that it won't include any duplicates in same or different classes and seemed reasonable.

Feature extraction was tested with word count vectorizer and TF-IDF vectorizer. There was no huge noticeable difference between these two methods. TF-IDF was chosen and it was used in all experiments with different parameter settings for n-gram range and stop words list. Table 8 shows how using different n-grams affected to accuracy of the classifiers. Grey box shows accuracy when using cross validation. Feature extraction is done with TF-IDF vectorizer function.

Table 8. Accuracy for different n-gram range.

Feature extraction (n-gram, stop)		None, None		(1, 2), None		(1, 3), None	
Feature selection		None		None		None	
Accuracy	k-NN	0.650	0.605	0.540	0.628	0.660	0.620
	NB	0.770	0.695	0.690	0.706	0.780	0.708
	RF	0.790	0.730	0.670	0.719	0.750	0.713

If using n-gram range, best result was achieved with Naïve Bayes and k-NN using n-gram range (1, 3). Random forest gives the best result when not using n-gram at all. This is not very good choice, because when analyzing Finnish tweets, it is better to use n-gram range (1, 2) or (1, 3), because Finnish words change rapidly their meaning if a word is related to a certain other words.

Another feature extraction parameter is a stop words list. Two different sizes stop words lists were tested and the best result was achieved using list where is 175 words. Using stop words affect also to the informative features. Table 9 shows how the stop words list

affected to accuracy where grey box indicates accuracy when using cross validation with 10 folds. Number in parentheses describes the amount of stop words. Table 10 shows some descriptive word types that are included in most informative features for Naïve Bayes.

Table 9. Accuracy for different size of stop words lists.

Feature extraction (n-gram, stop)		(1, 3), stop(0)		(1, 3) stop(1828)		(1, 3) stop(175)	
Feature selection		None		None		None	
Accuracy	k-NN	0.660	0.620	0.720	0.600	0.660	0.564
	NB	0.780	0.708	0.780	0.680	0.840	0.660
	RF	0.750	0.713	0.760	0.667	0.740	0.684

Table 10. Top 10 most informative feature types for Naïve Bayes.

Stop words = 175	-1	Curse words, disappointment, lack of understanding
	0	Words that does not include any sentiment
	1	Appreciation, supporting, affection

When feature extraction was used without any feature selection method, the accuracy of the classifiers was slightly lower when also using cross validation. Table 11 shows what happens to accuracy when feature selection methods were involved.

Table 11. Accuracy of the classifiers when using n-gram and stop words with feature selection methods.

Feature extraction		(1, 3) stop(175)		(1, 3) stop(175)		(1, 3) stop(175)		(1, 3) stop(175)	
Feature selection		Recursive Feature Elimination		Forward selection		Select K-Best (Chi-square, k=10)		Lasso	
Accuracy	k-NN	0.620	0.576	0.680	0.598	0.650	0.591	-	-
	NB	0.704	0.635	0.710	0.691	0.540	0.593	0.730	0.654
	RF	0.625	0.700	0.630	0.725	0.590	0.507	0.770	0.704

From feature selection methods Lasso gave the best results when used with Random forest. Naïve Bayes works best with forward selection or with none feature selection method. The best feature selection method for k-Nearest Neighbor was Forward selection. K-NN achieved good results when using no feature selector at all.

Figures 7, 8, and 9 shows the classification report with the confusion matrix for tested classifiers. Confusion matrix reports the number of false positives, false negatives, true positives and true negatives. This allows more detailed analysis than mere proportion of correct guesses (accuracy). Parameters for the classifier, training and testing time are also included in figures. The precision is the ratio that the classifier does not label as positive a sample that is negative [13]. The recall means the ability to find all the positive samples. F1-score is a weighted average of the precision and recall. Support describes the number of occurrences of each class.

k-NN

Training:
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
 metric_params=None, n_neighbors=10, p=2, weights='uniform')

train time: 5.431
 test time: 13.025
 accuracy: 0.600
 top 10 keywords per class:
 classification report:

	precision	recall	f1-score	support
-1	0.85	0.72	0.78	32
0	0.42	0.58	0.49	31
1	0.63	0.51	0.57	37
avg / total	0.64	0.60	0.61	100

confusion matrix:
 [[23 7 2]
 [4 18 9]
 [0 18 19]]

Figure 7. Classification report for k-NN.

In Figure 7, feature selection method for k-NN was dismissed because of the bad results, that classifier gave. Reason for this is caused by the learning data. Tests showed that the k-NN gives good results, if the learning data is specific and carefully selected for the new data or the training and test data is relatively small. K-NN accuracy varied the most during testing and was never a number one. Because other classifiers managed better, k-NN was bit left behind in the developing process for classification.

```

=====
Naive Bayes
=====
Training:
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
train time: 0.932
test time: 0.044
accuracy: 0.840
classification report:
      precision    recall  f1-score   support

     -1       0.89       0.89       0.89        37
      0       0.79       0.73       0.76        26
      1       0.82       0.86       0.84        37

 avg / total       0.84       0.84       0.84       100

confusion matrix:
[[33  1  3]
 [ 3 19  4]
 [ 1  4 32]]

```

Figure 8. Classification report for Multinomial Naïve Bayes.

Multinomial Naïve Bayes was the most accurate classifier for Finnish tweets. Naturally this was selected as a final classifier for the analysis. Naïve Bays did not need any feature selection, because the most informative features can be sorted from discrete features by their probability. In addition, Naïve Bayes works fine with forward selection.

```

=====
Random forest
=====
Training:
RandomForestClassifier(bootstrap=True, compute_importances=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, min_density=None, min_samples_leaf=1,
                        min_samples_split=2, n_estimators=1000, n_jobs=1,
                        oob_score=False, random_state=None, verbose=0)
train time: 890.792
test time: 0.322
accuracy: 0.740
top 10 keywords per class:
classification report:
      precision    recall  f1-score   support

     -1       0.90       0.81       0.85        32
      0       0.67       0.45       0.54        31
      1       0.68       0.92       0.78        37

 avg / total       0.75       0.74       0.73       100

confusion matrix:
[[26  4  2]
 [ 3 14 14]
 [ 0  3 34]]

```

Figure 9. Classification report for Random forest.

Classification report in Figure 9 does not include a feature selection. Figure 10 shows the report for Random forest with Lasso. These reports indicate that Random forest is more accurate when feature selection is involved. This result is based on the cross validation.

Random forest with L1-based feature selection

```

Training:
Pipeline(steps=[('feature_selection', LogisticRegression(C=100.0, class_weight=None,
intercept_scaling=1, penalty='l1', random_state=None, tol=0.0001)), ('class
criterion='gini', ...les_split=2, n_estimators=10, n_jobs=1,
oob_score=False, random_state=None, verbose=0))])
train time: 1.085
test time: 0.016
accuracy: 0.610
cross validation: 0.709115824108
classification report:
      precision    recall  f1-score   support

-1         0.54         0.85         0.66         26
 0         0.44         0.26         0.33         31
 1         0.76         0.72         0.74         43

avg / total         0.60         0.61         0.59         100

confusion matrix:
[[22  2  2]
 [15  8  8]
 [ 4  8 31]]

```

Figure 10. Classification report for Random forest with Lasso.

Most of the errors during the project were made in feature selection. From tested feature selection methods, Variance threshold was discarded right after trying. Discarding happened, because all of the features in the data have such a low variance, that feature selector removes them all. Variance threshold is better for unsupervised learning, because feature selector looks only at the features (X), not the desired outputs (y). Univariate feature selection method did not work either for the Twitter data. Chi-square and Select K-Best did not exceed to the same level as forward selection and Lasso. Forward and backward feature selection works quite well. Backward method demands more computation power than forward selection and it is clearly noticeable time difference (>20min).

Training time rose when using cross validation and especially when classifying with Random forest, using Lasso or RFE as a feature selection method. Table 12 shows some differences between training times. Training time was almost 50 minutes per test.

Table 12. *Classifiers training time with cross validation.*

Classifier	Training time
K-Nearest Neighbor	5.521
Naïve Bayes	0.826
Random Forest	878.736
Random Forest with Lasso	88.291
Naïve Bayes with Lasso	83.303

5. CONCLUSIONS

This study provides a description of a sentiment analyser for text data, using supervised machine learning algorithms. The purpose of the analyser is to mine sentiments from tweets and classify them into 3 classes; positive, negative and neutral. When millions of tweets travel in hyperspace at daily basis, manual analyzing is impossible. When using a machine to analyze tweets straight from the stream, information gain can be maximized.

Data science has been a hot topic and various studies about mining the Twitter data and analyzing the sentiments from content has been done before this work. Media industry has seen an opportunity in social media to create interactivity between brands and people. Mining the social media era is at its early phases and there are opportunities for developing analysis in many ways.

When training the classifier it is important to choose the right datasets. If datasets are too ambiguous, contain mixed sentiments or make comparisons, they are not ideal to be used for training. Using human annotated datasets as much as possible gives better results than automatically extracted examples. When annotating, it is important to remember that the probability of classifying a document as positive, negative or neutral is equal. Thus in the dataset the number of examples in each category should be equal.

Feature extraction can be done either using word count or TF-IDF vectorizer function. This study exploits TF-IDF, which also measures how important a word is. When making feature extraction using n-gram range and stop words list, accuracy can raise. Feature selection is another important part of the classification and the best method can be found by trial and error. When implementing feature selection method Lasso to Random forest classifier, it gave small increase to accuracy. Multinomial Naive Bayes reached better readings without any extra feature selection method. Random forest got better results for classification than Naive Bayes, when not using any n-gram or stop words list. These parameters were used anyway, because of the Finnish language structure, n-grams are needed.

Most of the previous studies about Twitter sentiment analysis have been done in English. These studies have tested and reported various text classification methods, where the reached accuracy level is <0.85 . This means that ≥ 15 percent of the test data is labeled in wrong class. This study differs in a way that the training data contains 2100 Finnish tweets. The best algorithm classified tweets by accuracy of 0.84. This accuracy can be received using Multinomial Naïve Bayes. This accuracy depends how the train-

ing and test data is divided and the analysator does not give at every time this accuracy. When using and relying cross validated values, true accuracy is present. By this assumption forward selection is the best feature selection method for all tested classifiers and gives cross validation 0.725 for Random forest. Getting 100 percent accuracy for classifying Twitter data automatically is impossible. For example context, sentiment ambiguity, and sarcasm have effect.

The study indicates that it is possible to extract sentiments, emotions and opinions from social media's data using machine learning algorithms. Accuracy 0.84 is almost as good as other studies has indicated so this project managed well. Mathematical patterns behind classifying algorithms can be really simple, naïve or they can be more complex trees.

With the analyzer created, tweets in Finnish can be now analyzed with supervised machine learning into classes positive, neutral and negative. This gives valuable information for example companies which want to know how people feel about them or media who wants to know how people react to a brand. Analysis can be also used for making a questionnaire in Twitter and analyse people's responses.

The best information source for sentiment analysis is academic papers. Choosing between different techniques is time consuming, and not every suggested technique will work well in every case. Some techniques work well only in specific domains and the quality of the results varies. When choosing a technique to be used in classification process, it is important to test different algorithms and not blindly choose some method from previous academic papers. Algorithms should work more accurate and efficient way than make things unnecessary complicated.

Algorithms and possibilities for classifiers and parameters are available in great amount. Data can have more preprocessing steps than this work comprises. These operations can include for example compressing words or creating a pattern to recognize sarcasm. Developing process can be continued to recognize sentiment also from other social media platforms and from images. One possibility is to use sentiment analysis in text-to-speech synthesis.

Phenomenon of the sentiment classification needs machine learning and natural language processing techniques to extract knowledge from the growing data. It stays important to extract irrelevant and redundant features away from the training data so the quality and cost of mining process stays appropriate.

REFERENCES

- [1] L. Breiman, Random Forests, University of California, January 2001, 33 p. Available: <http://oz.berkeley.edu/~breiman/randomforest2001.pdf>
- [2] T.G. Dietterich, Machine Learning, Oregon State University, 14 p. Available: <http://web.engr.oregonstate.edu/~tgd/publications/nature-ecs-machine-learning.pdf>
- [3] J. Gholamreza, T. Reza, P. Parvaneh, M. Ali, Hybrid Financial Analysis Model for Predicting Bankruptcy, British Journal of Economics, Vol.2(1), October 2011, 37-48 p. Available: <http://www.ajournal.co.uk/EFpdfs/EFvolume2%281%29/EFVol.2%20%281%29%20Article%204.pdf>
- [4] J. Han, M. Kamber, J. Pei, Data Mining Concepts and Techniques (Third edition), A volume in The Morgan Kaufmann Series in Data Management Systems, 2011, 5-9, 24-26, 327-386 p. Available: <http://www.sciencedirect.com/science/book/9780123814791>
- [5] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Data Mining, Inference, and Prediction, Second Edition, August 2008, 9-29, 305-317 p. Available: http://web.stanford.edu/~hastie/local.ftp/Springer/OLD/ESLII_print4.pdf
- [6] A. Jain, R. Duin, J. Mao, Statistical Pattern Recognition: A Review, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, NO. 1, January 2000, 4-37 p. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=824819>
- [7] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, 2013, 12, 38-40, 127 p. Available: <http://www-bcf.usc.edu/~gareth/ISL/ISLR%20First%20Printing.pdf>
- [8] C. Kaushik, A. Mishra, A Scalable, Lexicon based Technique for Sentiment Analysis, YMCA University of Science & Technology, International Journal in Foundations of Computer Science & Technology, Vol.4, No.5, September 2014, 35-43 p. Available: <http://arxiv.org/pdf/1410.2265.pdf>
- [9] M. Koppel, J. Schler, The Importance of Neutral Examples for Learning Sentiment, Bari-Ilan University, 2006, 1-9 p. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.9735&rep=rep1&type=pdf>

- [10] E. Loper, Natural Language Toolkit: Naive Bayes Classifiers, University of Pennsylvania, 2004, Available: <http://web.mit.edu/6.863/python/nltk-0.9.1.old/classify/naivebayes.py>
- [11] C. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008, 271-274 p. Available: <http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf>
- [12] A. Pak, P. Paroubek, Twitter as a Corpus for Sentiment Analysis and Opinion Mining, Université de Paris-Sud, 2010, 1320-1326 p. Available: [http://incc-tps.googlecode.com/svn/trunk/TPFinal/bibliografia/Pak%20and%20Paroubek%20\(2010\).%20Twitter%20as%20a%20Corpus%20for%20Sentiment%20Analysis%20and%20Opinion%20Mining.pdf](http://incc-tps.googlecode.com/svn/trunk/TPFinal/bibliografia/Pak%20and%20Paroubek%20(2010).%20Twitter%20as%20a%20Corpus%20for%20Sentiment%20Analysis%20and%20Opinion%20Mining.pdf)
- [13] Pedregosa et al. Scikit-learn: Machine Learning in Python, JMLR 12, 2011, 2825-2830 pp.
- [14] M.A. Russell, Mining the Social Web, O'Reilly Media, October 2013, 5-44 p.
- [15] M. Thelwall, K. Buckley, G. Paltoglou, Sentiment in Twitter events, Journal of the American Society for Information Science and Technology, 62(2), February 2011, 406-418 p. Available: http://www.researchgate.net/profile/Mike_Thelwall/publication/220435012_Sentiment_in_Twitter_events/links/0912f51333fad59559000000.pdf
- [16] M. Thelwall, Heart and Soul: Sentiment Strength Detection in the Social Web with SentiStrength, Statistical Cybermetrics Research Group, University of Wolverhampton, August 2013, 1-14 p. Available: <http://sentistrength.wlv.ac.uk/documentation/SentiStrengthChapter.pdf>
- [17] G. Valentine, T.G. Dietterich, Bias-variance analysis of Support Vector Machines for the development of SVM-based ensemble methods, Journal of Machine Learning Research 1, 2000, 1-48 p. Available: <http://web.engr.oregonstate.edu/~tgd/publications/jmlr-valentini-bv.pdf>
- [18] V. Vryniotis, 10 Tips for Sentiment Analysis projects, Blog on Machine Learning, Statistics & Software Development, September 2013, Available: blog.datumbox.com
- [19] S. Watanabe, Pattern Recognition: Human and Mechanical. New York: Wiley, 1985, 570 p.
- [20] Y. Whye Teh, Supervised Learning: Ensemble Methods, Random Forests, Department of Statistics, University of Oxford, 1-18 p. Available: <http://www.stats.ox.ac.uk/~teh/teaching/datamining/115a-RF.pdf>